Variational water-wave models and pyramidal freak waves

Floriane Marie Pauline Gidel

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

The University of Leeds

Department of Applied Mathematics

June 2018

Declaration

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.

The work in Chapter 2 of the thesis has appeared in publication as follows:

[57] F. Gidel, O. Bokhove and A. Kalogirou, *Variational modelling of extreme waves through oblique interaction of solitary waves: application to Mach reflection*, Nonlinear Processes in Geophysics, **24** (2017), 43-60, doi: 10.5194/npg-24-43-2017.

The original idea was given by O. Bokhove as an extension of his previous collaboration with A. Kalogirou [20]. Therefore, sections 2.2.2, 2.4.1 and 2.4.2 follow the method initially introduced in [20]. The other sections are directly attributable to the candidate's work. In addition, the candidate wrote the publication fully, with proofreading from A. Kalogirou, O.Bokhove, M. Kelmanson, G. Kapsenberg and T. Bunnik.

The work in Chapters 3 and 4 of the thesis has appeared in publication as follows:

[58] F. Gidel, O.Bokhove and M. Kelmanson, *Driven nonlinear potential flow with wave breaking at shallow-water beaches*, Proc. ASME 2017 36th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2017, **1** (2017).

The strategies presented in this paper were obtained through discussions and collaboration with O. Bokhove and M. Kelmanson. The candidate derived and implemented the mathematical and numerical models, and wrote the publication, with proofreading from O. Bokhove, M. Kelmanson, T. Bunnik and G. Kapsenberg.

The experiments in Chapter 5 were planned with and supervised by T. Bunnik and G. Kapsenberg.

À Guy Gidel,

et à ma famille, sans qui ce projet n'aurait pas pu aboutir

Acknowledgements

This thesis would not have been possible without the support of many great individuals, to whom I want to dedicate the first lines of this manuscript.

First, many thanks to my supervisors, Onno Bokhove and Mark Kelmanson, for their help and advice during the last four years. Onno, thank you for creating and leading this great project, and for having given me the opportunity to take part in it. I was privileged to experience such a rewarding and instructive work and that would not have been possible without your passion and infectious enthusiasm about research. This thorough project was the best way to begin my research career, so I sincerely thank you for your trust, your involvement and your guidance. Mark, thank you for your encouraging feedbacks that have been essential to keep me motivated and stimulated me to always do my best. Thanks also for your patience when correcting hundreds of times the same "Frenglish" typos; your insightful suggestions and advice have greatly improved the quality of this manuscript. More generally, thank you both for sharing your ambition and helping aim further than I could imagine. The PGR award is a great result of this ambition.

Many thanks also to my MARIN supervisors, Tim Bunnik and Geert Kapsenberg, for their great help during my secondment. Thanks for sharing your knowledge and expertise about the maritime industry and its challenges; our interesting discussions helped me to define the objectives of this thesis. I am especially grateful for your assistance before and during the experiments, which considerably improved the quality and impact of this thesis.

My sincere gratitude to David Hughes and Jennifer Ryan for assessing this work.

Special thanks also to Rainer Hollerbach for his precious help during the first half of my PhD. I am really grateful that you took the time to go through the details with me and helped me overcome my issues. Thanks also to Bulent Düz for his interest in my research and his insightful feedback on my code tutorials. My gratitude to Lawrence Mitchell for his precious help with Firedrake and the optimisation of the solvers, and with his feedback on the tutorials.

This PhD thesis would not have been possible without the financial support of the European Marie Skłodowska-Curie Actions (MSCA) Fellowship. Being a MSCA fellow was a chance in many aspects and I am really thankful for this support. Essentially, by combining academic and industrial experience, it enabled me to address all important stages of the modelling process; that is, the specification, the modelling, and the experimental validation.

I also thank the members of the School of Mathematics who, directly or indirectly, contributed to the smooth progress of this thesis. Thanks also to the "water team" for the nice meetings, during and after work, and, in particular, to Tomasz for sharing both the Leeds and MARIN adventures with me. My warmest thanks to Tom for his precious advice, his help and valuable support from first day to submission, the nice discussions, and, more importantly, all the beers!

On a personal note, I want to express all my love and gratitude to my parents, Dominique and Marie-Hélène, for their unconditional support. Thank you for giving me the chance to grow in the best conditions, for your help with everything I undertake, for comforting me in the difficult moments, for your patience and so much more. In addition, thanks Mum for proofreading parts of this thesis. I am also deeply grateful to my brothers, Pierre and Grégoire, for their invaluable support and humour over the years. Growing next to you was the best adventure and I am the proudest sister. Many thanks to Charlotte as well for being such a great sister(-in-law), and to my second family, les Giroux, for their precious support and all the great moments. Special thanks to my dearest friend, Anaïs, for her support, kindness, humour, advice, and even proofreading.

Thanks to all my cousins, uncles and aunts, for their encouragements, their interest in my research, their continued support, their advice, and for the lovely family reunions. I am so proud to be part of a such a great family! Thanks to my grandmothers, Paulette and Marie-Thérèse, for their love and unconditional support. I am so proud to have grandmothers who, while being over 90, Skype me when I am abroad, try to understand my research and to read my English publications, visit me in Netherlands, and always encourage me. Thank you both for being great examples.

Thanks to the most courageous, Pierre, for encouraging me to embark upon this project and to finish it. Most importantly, thank you for accepting all the constraints of my three years abroad, for your advice in the most difficult moments, for believing in me when I do not, and for your priceless patience. Thanks also to your family for their warm welcome.

I also want to address my warmest thanks to my friends for accompanying me on this journey despite living in another country. Special thanks to those who visited us in Leeds or Utrecht (or both!) for the great memories (Zoé, Clémentine, Marjory, Julie, Alice, Flo(flo), Sarah, Romain, Éric, Pado, Flo(fleur), Marie, Paulbib, Bozzo, les Lopez...). "Bisochat" to the fan club for all the good times that boosted me. I am deeply thankful to my cousin Margaux and my friends Floriane (both!), Julie, Marie for your continued encouragements; your support has been priceless. Thanks to Amélie and Clémence for all the great moments and laughs. Thanks to all the new friends I met during this adventure and who accompanied me during the last four years; I cannot cite you all, but special thanks to Charlotte, Pauline, Kaatje and Florian for the Dutch souvenirs, and to Mathilde for sharing both UK and Dutch memories!

I want to dedicate my last thoughts to my grandfathers, André Langrand and Guy Gidel, who cannot see the outcome of this project but would have been the proudest. Thank you for giving me your taste for science and for always being proud of me.

Abstract

A little-known fact is that, every week, two ships weighing over 100 tonnes sink in oceans [32], sometimes with tragic consequences. This alarming observation suggests that maritime structures may be struck by stronger waves than those they were designed to withstand. These are the legendary *rogue* (or *freak*) *waves*, *i.e.*, suddenly appearing huge waves that have traumatised mariners for centuries and currently remain an unavoidable threat to ships, and to their crews and passengers. Thus motivated, an EU-funded collaboration between the Department of Applied Mathematics (Leeds University) and the Maritime Research Institute Netherlands (MARIN) supported this project, in which the ultimate goal, of importance to the international maritime sector, is to develop reliable damage-prediction tools, leading to beneficial impact in terms of both safety and costs. To understand the behaviour of rogue waves, cost-effective water-wave models are derived in both deep and shallow water. Novel mathematical and numerical strategies are introduced to capture the dynamic air-water interface and to ensure conservation of important properties. Specifically, advanced variational Galerkin finite-element methods are used to provide stable simulations of potential-flow water waves in a basin with wavemakers and seabed topography, which allows reliable simulations of rogue waves in a target area. For optimised computational speed, wave absorption is considered with a beach on which waves break and dissipate energy. Robust integrators are therefore introduced to couple the potential-flow model to shallow-water wave dynamics at the beach. Experimental validation of the numerical tank is conducted at Delft University of Technology to ensure accuracy of the simulations from the wavemaker to the beach. The numerical tank is designed for subsequent use by MARIN to investigate the damage caused by rogue waves on structures in order to update maritime design practice and to ensure safety of ships, therefore leading to a competitive commercial advantage across Europe.

Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

“ Nothing in life is to be feared, it is only to be understood”, Marie Curie [11].

Mme Curie’s well-known assertion is arguably *not* the case for rogue waves (also called freak waves), which were feared throughout history; even since scientists have begun to understand them. Despite having traumatised, if not killed, sailors for centuries, these suddenly appearing, huge waves started to raise scientific attention only relatively recently, as late as the end of the 20th century. One of the first explorers to report such an event was Christopher Columbus who, in a letter to Castilian Sovereigns, during his third Voyage in 1498 [73], observed:

“He heard a terrible roaring from the south, and beheld the sea heaped up, as it were, into a great ridge or hill, the height of the ship, covered with foam, and rolling toward him with a tremendous uproar. His own ship was suddenly lifted up to such a height that he dreaded lest it should be overturned or cast upon the rocks, while another of the ships was torn violently from her anchorage. The crews were for a time in great consternation, fearing they should be swallowed up; but the mountainous surge passed on, and gradually subsided”.

A similar event was observed four centuries later, in 1826, by the naval officer Dumont d’Urville, whose testimony was believed to be folklore, in particular by the scientist François Arago [75]. Despite many other reports of similar events (see examples in [110]), it was indeed difficult for scientists to believe in the existence of such waves, which they could neither witness nor explain with existing knowledge. This scepticism has to be understood within the context of the 19th-century findings on water waves (see a chronology in [35]). At the time when d'Urville witnessed the 33m-high wave, knowledge on water waves was still very limited: his accident coincided with the first milestones on experimental and mathematical theory for water waves with, for instance, the book of Weber and Weber (1825) [141] and the essay of Cauchy (1827) on the initial-value problem for linearized water waves [26]. Later mathematical findings did not support mariners' stories either, since linear wave theory, first introduced by Airy [4] in 1841, predicts that wave heights follow a Gaussian distribution; as a consequence, in a sea state with a 10m-averaged wave height, a 30m-high wave would occur only once every 10 000 years; that is, much less frequently than reported by sailors.

In the 20th century, several instances of severe ship damage (*e.g.*, the USS Memphis (1916) [129], the Michelangelo (1966) [22], the Neptune Sapphire (1973) [110], the Wilstar (1974) [22], the Taganrogskiy Zaliv (1985) [110]) and losses of strong vessels (*e.g.*, the Wasatah (1909) [63], the SS Edmund Fitzgerald (1975) [13], the MS München (1978)[10]) have offered supportive evidence that has increased both the credibility of mariners and the curiosity of scientists. As a result, the first paper on freak waves was published by Draper in 1964 [39] who, building on his knowledge on ocean waves and the measurement of a 20m-high wave on a British Ocean Weather Ship, claimed that *"exceptionally high waves are not curious and unexplained quirks of Nature. Their occurrence can be calculated with an acceptable degree of precision"*. In the meantime, progress on water-wave theory, in particular with the work of Stokes [131, 132] on nonlinear gravity-wave dynamics, led to a second-order correction to the linear wave theory for crest-height prediction (*i.e.*, the distance between the depth at rest and the highest point of the wave). In 1980, Tayfun [134] indeed showed that, while wave heights are well predicted by the Gaussian distribution (or, as extended by Longuet-Higgins in 1952 [91], the Rayleigh distribution), wave crests are higher and better predicted with his second-order-correction model. On the 1st of January 1995, a 25.6m-high wave was measured at the Draupner offshore platform; that is, 2.25 times higher than the recorded averaged wave height in this area of the North sea [65]. The "New Year wave" was not only the evidence that such extreme and sudden waves exist, but also the long-awaited observation to launch research into rogue waves.

Since the “New Year wave” in 1995, improved technologies have enabled new and relatively frequent records of extreme waves, which are designated as rogue waves when at least twice as high as the characteristic wave height [65] (*i.e.*, the average height of the largest third of surrounding waves). For example, more than ten waves of height exceeding 25m were spotted in a three-week period (1996) in satellite SAR images [124, 88, 123], which confirmed that extreme waves are more frequent than predicted by either linear or second-order models. In addition, several passenger-ship accidents reported in the news media in the early 21$^{\text{st}}$ century (*e.g.*, the Bremen and the Caledonian star (2001, [90]), the Dawn (2005, [47])) and accidents at the coast (see examples in [38]) have enabled scientists to list and map witnessed rogue-wave events in order to study the origin of rogue waves (examples of inventory are found in [107, 38, 90, 41, 109]). As a result, we know today that these waves occur in both deep and shallow water [109] and do not always occur due to stormy weather, but also due to particular marine conditions such as: in areas where waves travel against a strong current (*e.g.*, in the Agulhas current [123, 97, 87, 112, 136, 137, 14]); in areas with varying bottom topography [128, 139, 60, 14]; and, in crossing seas [135, 27, 138, 123, 16] as in the case of the New-Year wave [2]. A thorough overview of current knowledge on rogue waves is given in [14]. While some of the aforementioned conditions may be circumvented (such as the Agulhas current), other conditions are neither fully understood nor predictable and can therefore not be avoided, thus regularly leading to new accidents (recent ones are the Canadian whale-watching-boat tragedy in 2015 [99] and the Jean Nicoli passenger ship in the Mediterranean sea in 2017 [46]). Expected to become increasingly frequent due to global warming [17, 18], the threat of these unpredictable rogue waves must not be neglected. Building safer maritime structures that would not sink in the case of an unfortunate encounter is therefore essential for the safety of crew and passengers.

Designing maritime structures able to resist extreme events has two major requirements. First, rogue-wave dynamics must be understood in order to be generated in experimental tanks, wherein wave-structure interactions can be tested. Reproducing rogue waves experimentally was the motivation of several studies in the last two decades (see, *e.g.*, the EXTREME SEAS and the MaxWave projects). This was, for example, achieved by means of temporal focusing (waves with different lengths travel at different speeds to meet in a target area) [111, 67, 29] and through the generation of a Peregrine breather [114, 28]. Experimental investigation of rogue-wave impact upon maritime structures has therefore been possible [30]. However, the reliability of such tests depends on the number of repeated measurements; design practice indeed requires averaged calculations of wave force over a large sample of measurements in order to reduce uncertainty [14]. The high cost of experiments (several thousand euros per day of testing) precludes the maritime industry from being able to provide a sufficient number of measurements, thereby limiting reliability of current data and, as a direct consequence, update of design practice. The second design requirement is the statistical study of the probability for a vessel to encounter a rogue wave. As explained in the previous paragraph, rogue waves are more frequent than predicted by the linear and second-order models, which therefore cannot be used in a practical sense to improve ship design. Another crest-height distribution, the *Weibull* distribution [142], was introduced by Forristall in 2000 [49] and has since been used by the maritime industry to design offshore structures [14]; however, it underestimates the occurrence of rogue waves. A way to estimate the frequency of the occurrence of freak waves is to conduct field measurements of the waves at sea. However, as highlighted in [14], limited storage usually enables the measurement of sea-surface time series of only 20 minutes, which is too short to obtain reliable statistical properties of rogue waves [15]. An alternative solution to experimental or field measurements is the use of numerical models that could simulate rogue-wave dynamics and realistic sea states. While some accurate water-wave and rogue-wave models exist (WASIM, OceanWave3D, ReFRESCO), their use is currently computationally limited by their high demands on simulation time, data storage and energy use. A solution to the aforementioned dynamical and statistical design-practice requirements would be to derive a cost-effective water-wave-simulation tool, in order to: determine statistical properties of rogue waves in a realistic sea state; study wavemaker motion resulting in wave-structure interactions in a target area; and, optimize or replace repeated experiments and large-scale simulations of rogue-wave dynamics. Deriving such a model is the aim of the SurfsUp project.

Finally, we note that only the first half of Mme Curie's quotation was used to open this chapter: its continuation

" Now is the time to understand more, so that we may fear less",

succinctly captures the spirit in which the remainder of this work is undertaken. To this end, *i.e.* with increased understanding in mind, we embark upon the objectives of the SurfsUp project and the goals of the thesis.

1.2 Objectives

1.2.1 The SurfsUp project

The SurfsUp project ("SurfsUP: Freak Waves and Breaking Wave Impact on Offshore Structures") is a European Industry Doctorate (EU EID) collaboration between the Maritime Research Institute of Netherlands (MARIN) and the department of Applied Mathematics at the University of Leeds. Funded by the Marie Curie actions, the partnership aims to use and expand mathematical knowledge about water waves to answer international maritime and industrial challenges. The three-year project comprises both the development of cost-effective water-wave models at the University of Leeds (18 months) and the experimental validation of the numerical simulations at MARIN (18 months). By combining mathematical and industrial expertise, the objective is to provide an added value to MARIN's design practice with a view towards increasing safety of maritime structures. Two major tasks are considered: first, the modelling of a cost-effective water-wave model in which rogue-wave dynamics can be reproduced and tested; and, second, the modelling of wave-structure interactions. In particular, coupling these two models will enable the estimation of rogue-wave loads on maritime structures. This thesis focuses on the first task, as detailed after a brief introduction of MARIN's activity and facilities.

1.2.2 The Maritime Research Institute of Netherlands

Founded in 1932, MARIN is a worldwide leader in hydrodynamic and nautical research and development whose aim is to provide innovative design solutions to build better, safer, cleaner, and more cost-effective ships and marine platforms. Their activity is shared between several departments, each focusing on specific targets (ships, offshore developments, trial and monitoring, maritime simulations, nautical center, and research and development). This thesis is part of the research and development activity and, more specifically, the “Basin waves and extreme waves” program that aims to expand the experimental and numerical knowledge about wave generation and wave modelling in the basins. For that purpose, several wave tanks are available at MARIN, equipped with wind and wave generators, seabed topography and absorbing beach, in order to reproduce realistic sea states. A specification and a description of the use of each wave tank is available on MARIN’s website [98]. In the next section, MARIN’s research and industrial requirements are considered in order to set the objectives of this thesis.

1.2.3 Aims of this thesis

MARIN has developed several simulation software packages for water-wave dynamics coupled with maritime structures. The most widely used, PARNASSOS [70] and ReFRESCO [140, 43], provide accurate determination of physical effects and full-scale predictions that are used from the initial design process through to the performance testing of built structures. However, the high accuracy of these simulations require expensive computational resources that limit their use. In particular, the update of consultancy practice for the design of safer structures in a rogue-wave environment requires the simulation of long time series due to the low frequency of occurrence of such waves. Existing wave models at MARIN would necessitate days, weeks or even months to provide expected results, as well as costly computational resources and storage. The main motivation of this thesis is therefore to develop a cost-effective water-wave model that can optimise both large-scale simulations of rogue waves and experimental tests in the basins.

In order to optimise large-scale simulations of rogue waves, the water-wave models must satisfy several criteria. First, computational speed must be ensured by means of an optimisation process. Second, conservation of important properties such as the mass and the energy must be maintained despite the various length scales involved (domain length, wavelength, wave height). Robust discretisation techniques will therefore need to be derived. Third, the extreme height of rogue waves and their great steepness must be captured by demonstrably stable numerical methods. Advanced implementation techniques will therefore be designed to ensure stability of rogue-wave simulations. Finally, with a view towards being used as a tool for design practice, accuracy must be proven. To ensure accuracy, verification with theory and experimental validation will be organised and conducted to ensure that models can be trusted.

To optimise experimental studies of rogue-wave damage on realistic maritime structures, an objective of this thesis is also to develop a numerical tank that matches the design of MARIN's basins. Water waves will therefore be simulated in a computational basin that contains wavemakers for the generation of the waves, with a seabed topography to enable both shallow- and deep-water rogue-wave simulations, and with a sloping beach to absorb the breaking waves. Specifically, the numerical tank will ensure reliable simulations of rogue waves in a target area, in order to optimise experimental wavemaker input in MARIN's basins. In addition, the mathematical model and numerical methods will be derived with a view towards the addition of a maritime structure in the computational basin, in order to facilitate the coupling between the numerical tank and the wave-structure interaction model developed as the second task of the SurfsUp project. The resulting advanced mathematical and numerical model can subsequently be used to measure several configurations of rogue-wave impacts upon a vessel or a wind turbine, thereby providing the information hitherto missing for the update of industrial consulting practice.

To better explain the building process of the numerical tank, some mathematical and numerical background on water-wave modelling is first recalled in the next section.

1.3 Water-wave modelling

1.3.1 Main characteristics

A wave is characterised by its height H_s, which is the distance between its trough (the lowest displacement of the surface) and its crest (the highest surface displacement), and by its wavelength λ, which is the distance between two consecutive crests or troughs. A wave travels at the phase velocity c_p, which can be visualised as the speed at which the crest travels, and is computed as the ratio between the angular frequency ω, and the wave number $\kappa = 2\pi/\lambda$. The angular wave frequency is defined by the dispersion relation, which depends on three main effects: surface tension, gravity and nonlinearity.

A *capillary wave* is a wave for which the surface-tension effect is predominant. This is the case for waves with very small wavelength (typically few millimetres), for which the cohesion between the water molecules at the surface is stronger than the adhesion between the water and air molecules. In this work, only wavelengths of at least several centimetres will be considered, in which case the surface-tension effects are negligible by comparison with the gravity effect.

Waves with predominant gravity effects are called *gravity waves*. This is the case for water waves, which result from the displacement of water from the state of rest due to an external force (the wind or a wavemaker motion for instance) or an internal disequilibrium (in stratified media for example), while gravity tries to restore the state of rest. For gravity waves, the dispersion relation is given by

$$\omega^2 = g\kappa \tanh(\kappa H), \tag{1.1}$$

where g is the constant of gravity acceleration ($g = 9.81 m/s^2$), and H is the fluid (water) depth at rest. The dispersion relation (1.1) enables one to highlight several properties of water waves. As the wavenumber κ is inversely proportional to the wavelength λ, Eq. (1.1) indicates that, at constant depth H, the phase speed of long waves is larger than that of short waves: that is, waves travel at different speeds depending on their wavelength. This phenomenon is called *dispersion*, and is involved in most wave-wave interactions; it is therefore an important property to consider when studying rogue-wave dynamics. In addition, two limiting systems can be derived from (1.1). First, when $H \to \infty$, the dispersion relation (1.1) becomes

$$\omega^2 = g\kappa, \tag{1.2}$$

so the phase speed $c_p = \sqrt{g/\kappa}$ depends on the wavelength, meaning that dispersion is involved in the wave dynamics. In particular, in this *deep-water limit*, which holds when $H > \lambda/2$, the velocity of the wave packet, called the *group velocity*, must be distinguished from the phase velocity. The group velocity $c_g = \dfrac{\partial w}{\partial \kappa}$ is the speed at which the envelope of the waves travels in space. In the deep-water limit, the wave group travels twice as slow as the waves contained within the group. An interesting property of the group velocity is that it corresponds to the speed at which energy is transported by the wave group [122]; it is therefore an indicator to consider when computing rogue-wave dynamics and loads. On the other hand, when depth becomes shallow, *i.e.*

when $H \to 0$, the dispersion relation may be expanded as a Taylor series to yield, at leading order,

$$\omega^2 = \kappa^2 gH + O(\kappa H^2). \tag{1.3}$$

This *shallow-water limit* is considered when the depth is shallower than $\lambda/20$, and leads to a phase speed $c_p = \sqrt{gH}$ that does not depend on the wavelength. Therefore, shallow-water sea states are non dispersive. As a consequence, the wave group travels at the same speed as the phase.

Since waves are non dispersive in the shallow-water limit, effects other than dispersion, such as nonlinearity, should be considered when generating rogue waves in shallow water. The nonlinear effect was highlighted by Stokes [131, 132], who noticed that, in some cases, ocean-wave profiles are non-sinusoidal: crests are steeper than troughs. In fact, when high waves (relative to the water depth or wavelength) interact with each other, the resulting waves are not a linear sum of the initial waves as in the linear theory. Instead, second- (or higher-) order terms are involved in the wave profile, leading to an increase of the wave steepness (*i.e.*, the ratio between its height and its wavelength). In addition, the nonlinear effects play a role in the phase and group velocities: in shallow water, high nonlinear waves travel faster than lower waves; in deep water, the dispersion relation (1.2) is still valid up to second order, but not for higher-order waves for which the wave velocity is increased by the wave amplitude (see [108] for a classification of water-wave theories).

Therefore, dispersion and nonlinearity are two wave properties to incorporate to our water-wave models in order to simulate rogue waves. The governing equations used to model water-wave dynamics are considered next.

1.3.2 Modelling wave dynamics

Fluids are often characterised by the evolution of their depth and velocity, through the conservation of mass (the *continuity* equation), the conservation of momentum (the equation of *motion*) and the conservation of energy. The ocean is often assumed to be incompressible and is therefore well described by the *Navier-Stokes* equations, which describe incompressible viscous fluids. In addition, far from solid boundaries, the viscous effects of water are often negligible compared to the inertial forces. The balance between inertial and viscous forces is quantified by the Reynolds number. For large Reynolds number, that is, for negligible viscosity, the Navier-Stokes equations are simplified to the incompressible *Euler* equations for inviscid fluids. These latter equations, together with appropriate boundary conditions, are often used to simulate water waves in large-scale simulations. However, solving such systems of equations is complex, can rarely be achieved using closed-form mathematics, and hence requires significant computational resources. As one objective of this thesis is to provide cost-effective water-wave simulations, further simplifications are considered. These are outlined in the next sections.

Potential-flow theory

Another characteristic of water waves can be used to reduce the number of unknowns in the equations. In water waves, the motion of particles is mostly translational. Therefore, the ocean can be assumed to be irrotational, meaning that the vorticity generation $\boldsymbol{\omega}$ is neglected. As a consequence, the velocity $\boldsymbol{u} = (u, v, w)$ is expressed as the gradient of a potential, the so-called *velocity potential*. This assumption reduces considerably the computational need since the equations are expressed in terms of one unknown, the velocity potential $\phi(x, y, z, t)$, instead of the three velocity components (u, v, w). Under the potential-flow theory, the continuity equation becomes

$$\nabla^2 \phi = 0,$$

which is the *Laplace* equation and which holds in the entire fluid domain. To be solved, this equation needs to be augmented with boundary conditions on the fluid surfaces. For instance, at a fixed wall, a Neumann boundary condition assumes that no flow occurs through the boundary; mathematically, this condition is expressed as $\nabla\phi \cdot \boldsymbol{n} = 0$, with $\nabla\phi$ being the velocity and $\boldsymbol{n}$ the outward normal to the surface, so that the outward normal velocity is null. A major difficulty in water-wave modelling is the boundary between water and air, called the *free surface* and defined by

$$z = h(x, y, t),$$

with z the vertical coordinate and h the total depth of water. The source of the extreme difficulty associated with a free surface is that it is *a priori* unknown and hence must itself be determined as an integral part of the solution process because its location is needed in order to apply two nonlinear *free-surface-boundary-conditions*. First, a *kinematic* boundary condition, that expresses that the boundary moves with the fluid through the material derivative:

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

where the first term indicates that the fluid surface is unsteady (*i.e.*, time-dependent), and the second and third terms indicate that the fluid is non-uniform (*i.e.*, space-dependent). Second, a *dynamic* boundary condition that is derived from the Euler equation for irrotational fluid; that is, the *unsteady Bernouilli equation of motion*. With the assumption that the water pressure at the free surface is the atmospheric pressure, the equation of motion for mean water depth at rest H_0 reads:

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

which is the conservation of energy equation for water waves; it indicates that the temporal evolution of the velocity potential (first term) depends on the kinematic energy (that is, the energy resulting from the wave motion, expressed by the second term) and the potential energy (that is, energy resulting from the water depth, expressed by the third term). The Laplace equation, augmented with the kinematic and dynamic boundary conditions at the free surface and no normal flow at the walls, is sufficiently accurate to model water waves, including rogue waves, since it includes both nonlinearity and dispersion. However, solving such equations is still challenging due to the moving nonlinear free surface, which, as mentioned above, is both a boundary and an unknown of the system of equations. Deriving implementation strategies to solve the three-dimensional potential-flow model numerically is therefore one of the challenges of this thesis.

Further simplifications, such as linearization or considering the shallow-water limit, can also be applied to simplify the equations or reduce the three-dimensional free-surface domain to a two-dimensional horizontal domain, on which horizontal boundaries are simplified. These assumptions are used in this thesis to model rogue waves in shallow water and wave breaking at the beach. The advantages and limits of the resulting models are introduced next.

Shallow-water model

When the depth at rest H is shallow relative to the wavelength λ, with factor $H < \lambda/20$, the vertical structure of the velocity is small by comparison with the horizontal velocity components. As a consequence, the three-dimensional domain with free surface can be reduced to a two-dimensional horizontal domain in which the solutions are depth-averaged. Another advantageous aspect of this so-called *shallow-water model* is that it can describe the free-surface water-wave dynamics in the shore zone where wave breaking occurs as a result of the nonlinear effects introduced in section 1.3.1: specifically, the wave crest, which is higher than the rest of the wave, travels faster and overlaps the rest of the wave. The shallow-water model captures the discontinuous breaking as hydraulic bores, in which the mass and momentum are conserved but energy dissipates. As a consequence, the vertical vorticity cannot be neglected and the potential-flow theory is not valid for shallow-water breaking waves. The *nonlinear shallow-water equations* therefore describe the evolution of the depth of water $h(x, y, t)$ (conservation of mass) and of the depth-averaged horizontal velocity $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v})$ (conservation of momentum); they are an efficient way to model wave-energy absorption at the beach. However, as explained in section 1.3.1, waves under the shallow-water limit are non dispersive, while dispersion is an essential rogue-wave property since it leads to wave-wave interactions. The shallow-water assumption may therefore be used to model wave breaking at beaches, but is too limited to model rogue waves in shallow water. Alternative models are presented next.

The Korteweg-de-Vries model

In 1834, J.S. Russel observed a surprising wave on the Union Canal near Edinburgh [125]:

"I was observing the motion of a boat which was rapidly drawn along a narrow channel by a pair of horses, when the boat suddenly stopped – not so the mass of water in the channel which it had put in motion; it accumulated round the prow of the vessel in a state of violent agitation, then suddenly leaving it behind, rolled forward with great velocity, assuming the form of a large solitary elevation, a rounded, smooth and well-defined heap of water, which continued its course along the channel apparently without change of form or diminution of speed. I followed

it on horseback, and overtook it still rolling on at a rate of some eight or nine miles an hour, preserving its original figure some thirty feet long and a foot to a foot and a half in height. Its height gradually diminished, and after a chase of one or two miles I lost it in the windings of the channel. Such, in the month of August 1834, was my first chance interview with that singular and beautiful phenomenon which I have called the Wave of Translation."

This kind of solitary wave of permanent shape and constant velocity has since been then called a *"soliton"* and has been widely studied. In particular, Korteweg and de Vries [85] described in 1895 the long-wave soliton dynamics by means of the so-called *Korteweg-de-Vries* equation, in which weak dispersion (that is, dispersion obtained by considering terms up to order $O(\kappa^3)$ in the shallow-water limit of the dispersion relation given in section 1.3.1) is balanced by weak nonlinearity (that is, a nonlinear term weighted by a coefficient of order $O(\epsilon)$ with $\epsilon \ll 1$). The weakly dispersive, weakly nonlinear soliton solutions will therefore be considered in this thesis to simulate rogue waves in shallow water. However, a limit of the KdV model is that it allows soliton propagation in one direction only. In order to increase wave heights through wave-wave interactions, a model that includes multidirectional wave propagation would be more consistent. While a quasi-two-dimensional extension of the KdV model exists, the so-called *Kadomtsev–Petviashvili (KP) equation* [76], this thesis instead considers the simulation of shallow-water rogue waves under the Benney-Luke approximation, whose advantages are introduced next.

Benney-Luke-type model

An alternative weakly-dispersive, weakly-nonlinear model can be derived from the potential-flow equations using the small-amplitude and long-wave scaling parameters introduced by Milewski and Keller [106] and Pego and Quintero [113]. The shallow-water limit is then obtained by means of a second-order expansion of the velocity potential near the seabed, which reduces the three-dimensional free-surface domain to the horizontal plane [20]. The resulting equations, hereafter called the *Benney-Luke-type* model by analogy with the equations of Benney and Luke [12], have the advantage of representing wave propagation in several horizontal directions, thus facilitating the set-up of wave-wave interactions compared to the KP model. In addition, the Benney-Luke-type model is closer to potential-flow theory and, therefore, less restrictive than the KP model. In this thesis, shallow-water rogue waves will therefore be simulated as soliton-type solutions of the Benney-Luke-type model.

1.3.3 Numerical modelling

Solving the above mathematical models analytically is interesting in order to get an exact solution of the equations. However, while some problem-specific exact soliton solutions of the KdV and KP equations are well known (see, *e.g.*, [84]), analytical resolution of nonlinear equations is in general too complex; proving existence and smoothness of the solution of the Navier-Stokes equations is for example part of the seven Millennium Prize Problems [37]. Instead, the equations can be solved by means of numerical methods that exploit computer resources to compute an approximate solution of a discretised version of the continuous model. In this thesis, most of the mathematical models will be solved numerically using the finite-element method, whose principle and advantages are explained next.

Numerical methods aim to split continuous equations into a finite number of discrete equations that can be solved by computers. Just as a picture made of pixels, the computational domain is discretised (that is, split) into small elements, called *finite elements*, on which the solution of the equations is averaged. These finite elements are carefully defined so that they cover the whole domain without overlapping each other. The set of finite elements is called the *mesh*. The elements are made of nodes (the *degrees of freedom*) and edges, and can take various forms depending on the problem to solve and on the geometry of the initial domain (*e.g.*, triangles, tetrahedron, or quadrilaterals). The number of the elements is called the *spatial resolution* and their size must be chosen to ensure *convergence* of the solution, that is, to minimize the error of the approximated solution: the smaller the element, the better the approximation (since the solution is averaged on a smaller volume). The ratio between spatial and temporal resolutions must be chosen to ensure *stability*; that is, numerical well-posedness. Galerkin showed that complicated solutions can be computed by superposing several simpler functions, the so-called *basis functions*. The solution in each element is the interpolation of the nodal values, called the *coefficients*, through the basis functions. The aim of the finite-element method is to compute the coefficient values, that is, the contribution of each basis function to the solution in the element and therefore, to the solution in whole domain. The decomposition into the sum of weighted basis functions enables one to write the initial equation in linear matrix form both within each element and over the whole domain. As the nodes of one specific element also contribute to the solution in the neighbouring elements, the matrix in the whole domain, called the *global matrix*, is computed by adding the contributions of each elementary matrix to the corresponding node. This process, which consists in mapping the nodes of each element (the *local* nodes) to the nodes of the mesh (the *global* nodes) is called *assembling*. Solving the matrix system then leads to the coefficient solutions on each node of the mesh. The solution in the whole domain is then computed by interpolating these coefficients with basis functions and appropriate boundary conditions. In summary, the finite-element method consists of four main steps: the *discretization* of the domain with finite elements to form a *mesh*; the derivation of linear equations in each element by choosing appropriate *basis functions*; the *assembling*, that is, the combination of each element equation to obtain the global equation; and, the application of boundary conditions to solve the global equation. A more advanced explanation of the finite-element method can be found in, *e.g.*, [72, 31, 59].

All the difficulty of numerical modelling lies in finding a good balance between the cost of the numerical model (that is, its complexity and computational need) and the accuracy of the numerical solution (that is, how different it is from the exact solution). One advantage of the finite-element method is that the matrices are sparse and therefore require minimal computational resources; this is a considerable advantage when developing cost-effective simulation tools. In addition, the method is flexible through the choice of the element shape and size to better map the domain geometry, as well as through the choice of the basis functions. Moreover, in this thesis, the implementation of the aforementioned finite-element-method steps is also facilitated by the use of *Firedrake*, an automated system that solves partial differential equations using the finite-element method. For example, meshing is done with the *Mesh()* function of Firedrake, which ensures non-overlapping and optimal referencing of the finite elements. The decomposition of the solution in terms of the basis functions and coefficients is then done internally, meaning that the equations can be directly implemented in space-continuous form, after choosing the basis functions from among a wide range of options. After providing initial solutions and boundary conditions, Firedrake nonlinear solvers can be used to linearise the nonlinear equations with Newton iterations and to solve the linearised system with a Krylov subspace method. Several parameters, such as the Krylov method, the preconditioning options, or the convergence threshold, can be changed in order to optimise the solvers and the computational cost. The solver also takes care of the assembling, which is computed in parallel for optimised cost. More generally, the mesh and solvers are built so that the Firedrake code can be executed in parallel without any additional changes. Firedrake is therefore of great help to both implement cost-effective water-wave simulations and to provide models that can easily be extended to future applications. Detailed documentation can be found in [120].

Accuracy of the numerical solution relies on the stability of the numerical solver (that is, the simulations do not blow up as a result of any inherent extreme physics) and its efficiency in conserving mass, momentum and energy as dictated by the mathematical models. To ensure these properties, the mathematical and numerical models of this thesis are derived from a variational approach, whose principle is explained next.

1.3.4 Variational approach

The models introduced in section 1.3.2 can be derived from a variational principle initially introduced by Luke [92], and subsequently modified by Miles [104]. This formulation describes incompressible and inviscid potential flows with a free surface through the variations of the pressure of the fluid integrated over the domain; the so-called *Lagrangian*. The variational principle states that these variations, which actually consist of the variations of the water depth h and velocity potential ϕ, must be equal to zero. The Lagrangian, obtained from Bernouilli's equations, can be expressed in a Hamiltonian form, which makes it attractive for numerical simulations. Gagarina *et al.* [52] showed that robust time integrators can be used to discretise Hamiltonian systems and to solve them with discontinuous Galerkin finite-element methods. These integrators preserve the structure of the Hamiltonian which is of high interest when modelling high-amplitude water waves (such as rogue waves) in large basins, since the mass, amplitude and energy must be conserved. These integrators also work for a non-autonomous Hamiltonian, i.e., Hamiltonian with explicit time dependence such as the external forcing due to a wavemaker, since they can be transformed into an autonomous variational principle using a change of variable [20]. The variational approach is thus relevant and adequate for our objective.

1.4 Thesis overview

The aims of this thesis, introduced in section 1.2.3, are addressed as follows. First, the case of rogue waves in shallow water is considered in Chapter 2 by simulating soliton interaction using the Benney-Luke-type model. As explained in section 1.3.2, this model enables the simulation of weakly-nonlinear and weakly-dispersive waves in shallow seas, wherein several fatal accidents have been observed in recent years [109]. To ensure accuracy of the model, the rogue-wave simulations are verified against the predictions of Miles [103, 105] for the resonant amplification of interacting solitons. These shallow-water predictions are also an important tool to calibrate our numerical model in order to reach the highest dynamical amplification obtained in the literature. The obtained Benney-Luke-type model, which is easier to solve than the full three-dimensional potential-flow model, also enables the validation of the consistency of our variational approach and numerical methods for the simulations of rogue waves.

Second, in Chapter 3, finite-depth to deep-water waves are considered using potential-flow theory in a tank with seabed topography. Modelling and implementation strategies are introduced to capture the dynamical free surface of the nonlinear unbroken waves generated by a piston wavemaker. In addition, the wavemaker motion is tuned to generate a rogue wave from the dispersion effect based on the experimental data provided by MARIN [24]. In order to ensure the cost efficiency demanded by the maritime industry, the computational performance of the numerical model is optimised and the cost/accuracy balance is tested for two temporal integrators. With this potential-flow model and the Benney-Luke-type model of Chapter 2, rogue waves can be simulated in a target area and tested in both deep- and shallow-water sea states.

In Chapter 4, the numerical tank required by MARIN to test wave-structure interactions is constructed from the deep-water potential-flow model. In order to avoid wave reflection and disturbance of the target area, the vertical wall at the end of the deep-water basin of Chapter 3 is replaced by an absorbing beach on which waves lose energy when breaking. As explained in section 1.3.2, wave breaking cannot be modelled when using potential-flow theory, and the shallow-water equations should be solved instead. The main challenge of the numerical tank is therefore to stably and accurately couple the nonlinear potential-flow model in the deep-water area to the nonlinear shallow-water equations at the beach. To address this challenge, the equations are coupled variationally and the numerical strategies introduced and validated in Chapter 3 are applied to the coupled system. To verify consistency of the model, bidirectional conservation of energy from deep to shallow water and from shallow to deep water is verified.

The accuracy of the numerical tank is then evaluated in Chapter 5 by means of an experimental validation conducted at Delft Technical University (TUD). Various wave profiles are tested in order to ensure the efficiency of the numerical tank when modelling realistic sea states. In addition, the comparison between numerical simulations and experimental data aims to highlight the limits of the numerical tank with a view towards future improvements.

These future improvements, as well as the use of the present shallow-water (Chapter 2), deep-water (Chapter 3) and numerical-tank (Chapter 4) models are also facilitated by detailed Firedrake-implementation tutorials provided in Chapter 6. The instructions aim to not only help the maritime industry to use the water-wave models as an optimisation tool for large-scale simulations and experiments, but also to provide implementation strategies that can be further developed to address future constraints.

Finally, a summary of the models, their applications and limits are discussed in the conclusions.

Chapter 2

Rogue-type waves in shallow water: the example of solitary-wave interactions

2.1 Introduction

Offshore structures such as wind turbines, ships and platforms are designed to resist loads and stresses applied by winds, currents and water waves. These three factors can cause damage or destroy these structures when their effect is underestimated. Designers and engineers must take into account the effect of not only each of these phenomena separately, but also their interaction, which can increase their adverse effects. In this work, we focus on the impact of extreme waves created from the propagation of an obliquely incident solitary wave along the side of a ship (a wave–structure interaction) or its impact with another identical obliquely incident wave (a wave–wave interaction). These two cases are mathematically equivalent since reflection at a rigid wall (represented here by the ship's side) is modelled through the boundary condition of no normal flow at the wall, which is equivalent to the intersection of two identical waves travelling in opposite directions, in which case a virtual wall is formed. The study of extreme, freak or rogue waves resulting from reflection at a wall or interaction of waves has spawned different theories in the last 50 years, some of which are subsequently reviewed.

The objective of the present work is to simulate rogue-type waves in shallow water by applying a theory first introduced by Miles [105, 103] for the resonant interaction of obliquely interacting solitons. Miles first derived an analytical solution of the Korteweg-de Vries (KdV) equation of motion in the case of obliquely incident solitons [105]. By letting the phase shift between the incident solitons tend to minus infinity, he then obtained the resonant limit of the obliquely incident soliton solution [103], from which he derived resonance conditions on the incident solitons' wavenumbers and circular frequencies. His results enabled him to explain experimental results of Perroud [117], who observed that regular reflection of an obliquely incident solitary wave upon a rigid wall is not possible for sufficiently small angles of incidence. For a specific range of angle of incidence φ_{i} and scaled amplitude a_{i} of the wave, the reflection of the soliton may instead result in three wave fronts: the incident and reflected waves (of respective amplitudes a_{i} and a_{r}), as well as a Mach stem wave (of amplitude a_{w}) propagating along the wall with an increasing length. This kind of reflection is called "Mach reflection" and is illustrated in Fig. 2.1.

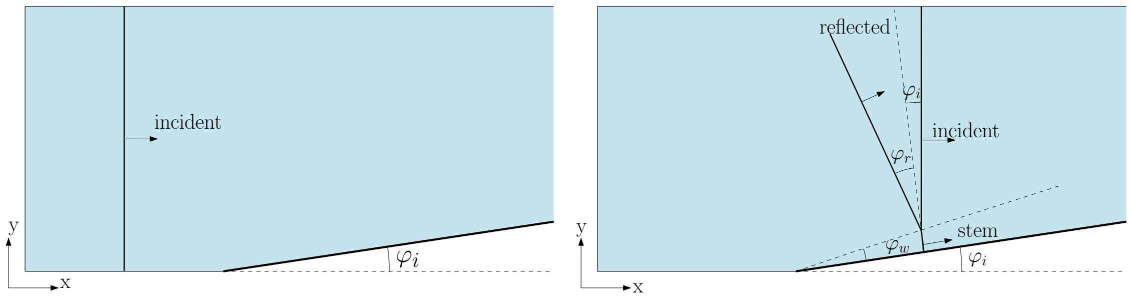


Figure 2.1: Left: top view of a channel containing an incident solitary wave propagating in the x direction with amplitude a_{i}. The side wall is oblique and makes an angle φ_{i} with the x direction. Right: top view of the reflection pattern when the incident wave impacts upon the wall. The pattern is composed of three waves: (1) the incident wave, (2) a reflected wave of amplitude a_{r} that forms an angle φ_{r} with the angle perpendicular to the wall, and (3) a Mach stem wave propagating along the wall with amplitude a_{w} and an angle φ_{w} with the wall.

Miles [103] showed that Mach reflection holds in the case of small-but-finite wave amplitude, shallow-but-finite water depth, and weak nonlinearity, that is,

$$\varphi_{\mathrm{i}}^2 = O(\epsilon), \quad a_{\mathrm{i}} = O(\epsilon), \quad \text{for any } \epsilon \ll O(1). \tag{2.2}$$

The resonance conditions obtained by Miles [103] on the solitons' wavenumbers and circular frequencies enabled the prediction of the amplitude and direction of propagation of each wave front, based on an interaction parameter, that he defined as

$$\kappa = \frac{\varphi_{\mathrm{i}}}{\sqrt{3a_{\mathrm{i}}}}. \tag{2.3}$$

The most important observation of Miles [103] is the transition at $\kappa = 1$ from a regular reflection ($\kappa \geq 1$) to a Mach reflection ($\kappa < 1$), which has led to the following definition of the stem-wave amplification,

$$\alpha_{\mathrm{w}} = \begin{cases} \dfrac{4}{1+\sqrt{1-\kappa^{-2}}}, & \text{for } \kappa \geq 1, \\ (1+\kappa)^2, & \text{for } \kappa < 1, \end{cases} \tag{2.4}$$

so that $\alpha_{\mathrm{w}} = a_{\mathrm{w}}/a_{\mathrm{i}}$ is the quotient of the stem-wave and incident-wave amplitudes. Equation (2.4) shows that at the transition point where $\kappa = 1$ the stem wave may grow up to 4 times the amplitude of the incident wave, leading to extreme loading on offshore structures. The aim of the present study is to develop a (numerical) model that can accurately simulate the evolution of the stem wave so that the distance and direction of propagation required to reach the 4-fold amplitude can be estimated. A challenging aspect is that it takes a long time and large distance of propagation before the stem wave reaches its maximum amplitude, which was a limiting factor in previous experimental and numerical studies. Kodama *et al.* [84] extended Miles' theory to the Kadomtsev–Petviashvili (KP) limit, in which the assumptions are

$$\frac{a_0}{H_0} = O(\epsilon), \qquad \left(\frac{H_0}{\lambda_0}\right)^2 = O(\epsilon), \qquad \tan^2\varphi_{\mathrm{i}} = O(\epsilon), \qquad \epsilon \ll O(1), \tag{2.5}$$

where H_0, a_0 and λ_0 are the water depth, the wave amplitude and wavelength, respectively. While the KP limit (2.5) still considers shallow-but-finite depth and small-but-finite amplitudes, the main difference with Miles' theory concerns the condition on the angle φ_{i}. Yeh *et al.* [146] explained that, in contrast to Miles' theory, wherein the soliton propagates in one direction only (the Korteweg–De Vries – KdV – limit), the KP limit assumes a quasi-two-dimensional approximation, and therefore the condition $\tan^2\varphi_{\mathrm{i}} = O(\epsilon)$ cannot be simplified to $\varphi_{\mathrm{i}}^2 = O(\epsilon)$ as in Miles' assumptions. The quasi-two-dimensional KP soliton is not a solution of the KdV equation, but it can be transformed to an asymptotic KdV soliton via some manipulations detailed in [146]. However, the width of the obtained KdV soliton is proportional to

$$\sqrt{\frac{a_{\mathrm{KP}}}{\cos^2 \varphi_{\mathrm{i}}}}, \tag{2.6}$$

with a_{KP} the scaled amplitude of the initial KP soliton, and it therefore depends on the angle φ_{i}. This is physically unrealistic since the KdV soliton should have the same shape whatever its direction of propagation. For this reason, Yeh *et al.* [146] brought a "high-order correction" to the solution, setting the amplitude of the KdV soliton to be

$$a_{\mathrm{KdV}} = \frac{a_{\mathrm{KP}}}{\cos^2 \varphi_{\mathrm{i}}}, \tag{2.7}$$

so that its width depends on its amplitude a_{KdV}, but not on any angle. Taking this into account, they slightly modified the definition (2.3) of the interaction parameter κ to

$$\kappa = \frac{\tan \varphi_{\mathrm{i}}}{\cos \varphi_{\mathrm{i}} \sqrt{3 a_{\mathrm{i}}}}, \tag{2.8}$$

where $a_{\mathrm{i}} = a_{\mathrm{KdV}}/H_0$ is the scaled amplitude of the incident wave, leading to what we will hereafter identify as the "modified Miles' theory" for the expected stem-wave amplification:

$$\alpha_w = \begin{cases} \dfrac{4}{1+\sqrt{1-\kappa^{-2}}}, & \text{for } \kappa \geq 1, \\ (1+\kappa)^2, & \text{for } \kappa < 1, \end{cases} \tag{2.4}$$

$$\text{with} \quad \kappa = \frac{\tan \varphi_i}{\cos \varphi_i \sqrt{3 a_i}}. \tag{2.8}$$

Using this modified interaction parameter (2.8) in Eq. (2.4), they found much better agreement between previous numerical simulations (*eg.* [50, 133]) and modified Miles' theory. Moreover, Kodama *et al.* [84] showed that the stem wave resulting from the interaction of two solitary waves with small incident angles is an exact solution of the KP equation. Solving this KP equation, they could describe the exact solution depending on the angle of incidence and the amplitude of the initial waves, and validate their theory with numerical simulations [84, 89]. Both the amplitude and length of the stem wave indeed followed their predictions in the case of regular and Mach reflection. The numerical scheme could not simulate the highest amplitudes that Miles predicts for $\kappa \approx 1$. Recently, Ablowitz and Curtis [1] studied Mach reflection for the Benney–Luke approximation, showing that, in that case, modified Miles' theory applies asymptotically, leading to amplifications of up to 3.9.

The purpose of the present work is to derive and apply a stable numerical scheme able to estimate the solution over a long distance of propagation, in order to model high-amplitude waves and to confirm the transition from regular to Mach reflection happening for $\kappa \approx 1$. We develop a model similar to the one of Benney and Luke [12], which is an asymptotic approximation of the potential-flow equations for small-amplitude and long waves. Whilst it has the advantage of conserving both the nonlinear and dispersive properties of the waves (essential to the modelling of a freak wave, for instance), it does not require a mesh moving vertically with the free surface since the model is reduced to the horizontal plane. Pego and Quintero [113] derived these modified Benney–Luke equations and Bokhove and Kalogirou [20] used them to simulate a soliton splash resulting from a wave running in a restricted channel. Their simulations were in reasonably good agreement with experiments, which confirms that the Benney–Luke approximation is an accurate model of water waves. The present approaches are necessary to determine how, in future work, we can impose the line solitons on the wavemakers to generate a 4-fold amplified wave in the middle of a wave basin and measure its impact on offshore structures. The variational technique used in the present approach enables us to express the equations as a Hamiltonian system to which robust time integrators can be applied [62, 52]. The space and time Galerkin finite-element method used to discretize the present model ensures the overall conservation of mass, energy and momentum, which are essential in the high-amplitude and long-distance propagating waves studied here.

The remainder of this chapter is organized as follows: the modified Benney–Luke-type model is derived in section 2.2 from the variational principle for an inviscid and incompressible fluid introduced by Luke [92] in the potential-flow approximation, using the small-amplitude and small-dispersion scaling of Pego and Quintero [113]. In order to apply modified Miles' theory and verify our numerical results against Kodama's exact solution, the KP limit is obtained from the Benney–Luke approximation, leading to a new variational principle for KP. A careful scaling is then defined in section 2.3 to obtain an asymptotic soliton solution of our present model, based on the exact solution of the KP equation from Kodama *et al.* [84]. The corresponding interaction parameter is consequently derived, leading to another version of modified Miles' theory (Eqs. 3 and 7), later used to compare our numerical simulations with respect to Miles' expectations. The finite-element method is then used in section 2.4 to discretize the equations in space together with the second-order Störmer–Verlet temporal scheme that ensures stable simulations. Results are finally discussed and compared to the expectations in section 2.5.

2.2 Water-wave model

2.2.1 Introduction

Our water-wave model is derived using a variational approach that ensures conservation of mass, momentum and energy. In a basic sea state with extreme waves, these conservation properties are essential given the different length scales involved. Starting from Luke's variational principle for an inviscid fluid with a free surface [92], a model similar to the one derived by Benney and Luke [12] for small-amplitude and long waves is obtained. The (numerical) method developed by Bokhove and Kalogirou[20] is used to derive the relevant variational principle for our Benney–Luke model. This asymptotic model conserves the nonlinear and dispersive properties of the sea waves, which enables comparison with the Kadomtsev–Petviashvili (KP) model for which the modified Miles' theory as expressed in Eqs. (2.4) and (2.8) applies.

2.2.2 From Luke's variational principle to the Benney–Luke set of equations

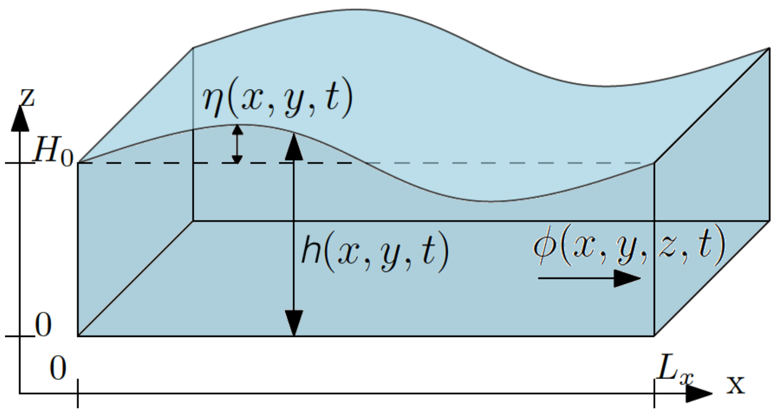


Figure 2.2: Three-dimensional water-wave domain with rest depth H_0, velocity potential $\phi(x,y,z,t)$, total depth $h(x,y,t)$ and free surface deviation $\eta(x,y,t)$.

Water-wave equations are often adequately described by the potential-flow approximation. In the absence of vorticity, the fluid velocity $\mathbf{u} = (u_x, u_y, u_z)$ can be expressed as the gradient of the so-called velocity potential $\phi(x, y, z)$, such that $\mathbf{u} = \nabla\phi$. The deviation from the surface at rest H_0 is defined by $\eta(x, y, t)$ so that the total depth $h(x, y, t)$ can be expressed as $h(x, y, t) = H_0 + \eta(x, y, t)$ (cf. Fig. 2.2). We consider a flat seabed lying at $z = 0$, with vertical walls at $\partial\Omega_{\mathrm{b}}$, where Ω_{b} is the horizontal plane of the bed coordinates $\Omega_{\mathrm{b}} = \{0 \le x \le L_x, 0 \le y \le L_y\}$. Luke [92] described an inviscid and incompressible fluid with a free surface in the potential-flow approximation through the following variational principle:

$$0 = \delta \int_0^T \int_{\Omega_{\mathrm{b}}} \int_0^{H_0+\eta(x,y,t)} \left[\partial_t\phi + \frac{1}{2}|\nabla\phi|^2 + \frac{1}{2}(\partial_z\phi)^2 + g(z - H_0) \right] \mathrm{d}z\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}t, \qquad (2.9)$$

where g is the acceleration of gravity. The gradient ∇ is defined on Ω_{b} only, such that $\nabla = (\ \partial_x\ ,\ \partial_y\ )^T$ is the horizontal gradient. The velocities at the walls and seabed are assumed to be zero, that is, $\mathbf{n} \cdot \nabla\phi = 0$ on $\partial\Omega_{\mathrm{b}}$, with $\mathbf{n}$ the outward horizontal normal and $\partial_z\phi = 0$ at $z = 0$. The boundary conditions at the free surface $z = h$ and the equations of motion in the domain Ω are obtained from Eq. (2.9) as

$$\nabla^2\phi + \partial_{zz}\phi = 0 \qquad \text{in } \Omega, \qquad (2.10a)$$

$$\partial_t\eta + \nabla\phi \cdot \nabla\eta - \partial_z\phi = 0 \qquad \text{at } z = h, \qquad (2.10b)$$

$$\partial_t\phi + \frac{1}{2}|\nabla\phi|^2 + \frac{1}{2}(\partial_z\phi)^2 + g\eta = 0 \qquad \text{at } z = h, \qquad (2.10c)$$

$$\mathbf{n} \cdot \nabla\phi = 0 \qquad \text{on } \partial\Omega_{\mathrm{b}}, \qquad (2.10d)$$

$$\partial_z\phi = 0 \qquad \text{at } z = 0. \qquad (2.10e)$$

The amplitude parameter $\epsilon = a/H_0 \ll 1$, with a the amplitude of the waves, and the small dispersion parameter $\mu = (H_0/\lambda_0)^2 \ll 1$, with λ_0 the horizontal wavelength, have been introduced by Milewski and Keller [106] and Pego and Quintero [113] to scale Eq. (2.9). The scaled variational principle is

$$0 = \delta \int_0^T \int_{\Omega_{\mathrm{b}}} \left\{ \int_0^{1+\epsilon\hat{\eta}} \left[\epsilon\partial_{\hat{t}}\hat{\phi} + \frac{\epsilon^2}{2}|\hat{\nabla}\hat{\phi}|^2 + \frac{1}{2}\frac{\epsilon^2}{\mu}(\partial_{\hat{z}}\hat{\phi})^2 \right] \mathrm{d}\hat{z} + \frac{1}{2}\epsilon^2\hat{\eta}^2 \right\} \mathrm{d}\hat{x}\, \mathrm{d}\hat{y}\, \mathrm{d}\hat{t}, \qquad (2.11)$$

where

$$\begin{aligned} \hat{x} &= \frac{\sqrt{\mu}}{H_0}x, \qquad \hat{y} = \frac{\sqrt{\mu}}{H_0}y, \qquad \hat{z} = \frac{1}{H_0}z, \qquad \hat{t} = \frac{\sqrt{gH_0\mu}}{H_0}t, \\ \hat{\eta} &= \frac{1}{\epsilon H_0}\eta \qquad \text{and} \qquad \hat{\phi} = \frac{\sqrt{\mu}}{\epsilon H_0\sqrt{\epsilon H_0}}\phi. \end{aligned} \tag{2.12}$$

From now on, the hats on the variables introduced in Eq. (2.12) are omitted.

The scaling (2.12) focusses on small-amplitude long waves. The relative sizes of ϵ and μ must be set depending on the balance between the nonlinear and dispersive effects. To satisfy conditions (2.5) introduced by Kodama *et al.* [84], ϵ and μ must satisfy $\mu = O(\epsilon)$. Similarly, the original Benney–Luke equations [12] assume $\epsilon = \mu$, meaning that the nonlinear and dispersive effects are balanced.

To obtain the Benney–Luke–type model in [20], the velocity potential ϕ is expanded in terms of the seabed potential $\phi(x, y, 0, t) = \Phi(x, y, t)$ and the dispersion parameter μ:

$$\phi(x, y, z, t) = \Phi(x, y, t) + \mu\Phi_1(x, y, z, t) + \mu^2\Phi_2(x, y, z, t) + O(\mu^3). \tag{2.13}$$

Combining the expansion (2.13) with the system of Eq. (2.10) and retaining terms up to second order, Eq. (2.13) becomes (see [20], for details)

$$\phi = \Phi - \frac{\mu}{2}z^2\Delta\Phi + \frac{\mu^2}{24}z^4\Delta^2\Phi + O(\mu^3). \tag{2.14}$$

Substituting Eq. (2.14) into the variational principle (2.11) and retaining terms up to order $O(\epsilon^2\mu, \epsilon^3)$ yield the variational principle under the Benney–Luke approximation [20]

$$0 = \delta\int_0^T\int_{\Omega_\mathrm{b}}\left[\eta\partial_t\Phi + \frac{\mu}{2}\nabla\eta\cdot\partial_t\nabla\Phi + \frac{1}{2}(1+\epsilon\eta)|\nabla\Phi|^2 + \frac{\mu}{3}(\Delta\Phi)^2 + \frac{1}{2}\eta^2\right]\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t. \tag{2.15}$$

Arbitrary variations in both Φ and η, together with boundary conditions $\mathbf{n}\cdot\nabla\Phi = 0$ and $\mathbf{n}\cdot\Delta\nabla\Phi = 0$ at $\partial\Omega_\mathrm{b}$, lead to the Benney–Luke equations

$$\delta\eta \;:\; \partial_t\Phi - \frac{\mu}{2}\partial_t\Delta\Phi + \frac{\epsilon}{2}|\nabla\Phi|^2 + \eta = 0, \tag{2.16a}$$

$$\delta\Phi \;:\; \partial_t\eta - \frac{\mu}{2}\partial_t\Delta\eta + \nabla\cdot((1+\epsilon\eta)\nabla\Phi) - \frac{2}{3}\mu\Delta^2\Phi = 0. \tag{2.16b}$$

Equations (2.16) will be solved numerically as explained in section 2.4. However, to test our Benney–Luke model on modified Miles' theory (Eqs. 3 and 7), it must first be compared to the KP theory for which Kodama *et al.* [84] have shown that modified Miles' theory holds.

2.2.3 From the Benney–Luke set of equations to the Kadomtsev–Petviashvili equation

In [20], Bokhove and Kalogirou introduced a scaling to derive the Korteweg–de–Vries equations from the Benney-Luke equations. Similarly, the Kadomtsev–Petviashvili equation for small-amplitude solitons can be derived from the Benney–Luke variational principle (2.15) and Eq. (2.16) through the transformations

$$X = \sqrt{\frac{\epsilon}{\mu}}(x-t), \qquad Y = \frac{\epsilon}{\sqrt{\mu}}y, \qquad \tau = \epsilon\sqrt{\frac{\epsilon}{\mu}}t, \qquad \Psi = \sqrt{\frac{\epsilon}{\mu}}\Phi \qquad \text{and} \quad \eta = \eta. \tag{2.17}$$

In order to apply the high-order correction arising from the quasi-two-dimensionality of the KP soliton [146] and introduced in Eq. (2.8), the order of ϵ will be set to $O(\epsilon) = O(\sqrt{\mu})$ in the numerical simulations so that $O(Y) = O(y)$ in Eq. (2.17). As a consequence, the nonlinear effect will be stronger than the dispersion effect, which is consistent with the definition of shallow–water rogue waves. Substituting scalings (2.17) into Eq. (2.16a), η can be expressed from Ψ as

$$\eta = \Psi_X - \epsilon\Psi_\tau - \frac{\epsilon}{2}\Psi_{XXX} - \frac{\epsilon}{2}\left(\Psi_X\right)^2 - \frac{\epsilon^2}{2}\left(\Psi_Y\right)^2 + \frac{\epsilon^2}{2}\Psi_{\tau XX} - \frac{\epsilon^3}{2}\Psi_{XYY} + \frac{\epsilon^3}{2}\Psi_{\tau YY}. \tag{2.18}$$

Substituting Eq. (2.17) into the transformed variational principle (2.15) yields

$$\begin{aligned} 0 = \delta\int_0^T\int_{\Omega_b}\Big[&\eta\left(\epsilon\Psi_\tau - \Psi_X\right) + \frac{\epsilon}{2}\eta_X\left(\epsilon\Psi_{\tau X} - \Psi_{XX}\right) + \frac{\epsilon^2}{2}\eta_Y\left(\epsilon\Psi_{\tau Y} - \Psi_{XY}\right) \\ &+ \frac{1}{2}(1+\epsilon\eta)\left(\left(\Psi_X\right)^2 + \epsilon\left(\Psi_Y\right)^2\right) + \frac{\epsilon}{3}\left(\left(\Psi_{XX}\right)^2 + \epsilon^2\left(\Psi_{YY}\right)^2\right) + \frac{1}{2}\eta^2\Big]\,\mathrm{d}X\,\mathrm{d}Y\,\mathrm{d}\tau. \end{aligned} \tag{2.19}$$

Subsequent elimination of η using Eq. (2.18) and truncation to $O(\epsilon^2)$ gives the variational principle for KP in terms of $\eta \approx \Psi_X$:

$$0 = \epsilon\delta\int_0^T\int_{\Omega_b}\left[\Psi_X\Psi_\tau + \frac{1}{2}(\Psi_X)^3 - \frac{1}{6}(\Psi_{XX})^2 + \frac{1}{2}(\Psi_Y)^2\right]\mathrm{d}X\,\mathrm{d}Y\,\mathrm{d}\tau \tag{2.20a}$$

$$= \epsilon\int_0^T\int_{\Omega_b}\delta\Psi\left[-2\Psi_{X\tau} - 3\Psi_X\Psi_{XX} - \frac{1}{3}\Psi_{XXXX} - \Psi_{YY}\right]\mathrm{d}X\,\mathrm{d}Y\,\mathrm{d}\tau. \tag{2.20b}$$

Note that we consider an infinite plane, with Ψ vanishing at the boundaries $|X,Y| \to \infty$, such that the boundary terms arising from the integration by parts vanish in Eq. (2.20b). Since $\delta\Psi$ is arbitrary, the variational principle (2.20) yields the following equation for the leading-order scaled potential Ψ:

$$2\Psi_{X\tau} + 3\Psi_X\Psi_{XX} + \frac{1}{3}\Psi_{XXXX} + \Psi_{YY} = 0. \tag{2.21}$$

From Eq. (2.18), at leading order in $O(\epsilon)$, η can be expressed as $\eta = \Psi_X$ and, therefore, taking the partial derivative of Eq. (2.21) with respect to X leads to the KP equation for η:

$$\left[2\eta_\tau + 3\eta\eta_X + \frac{1}{3}\eta_{XXX}\right]_X + \eta_{YY} = 0. \tag{2.22}$$

A solution of the KP Eq. (2.22) is found by substituting the following soliton solution ansatz, the form inspired by Eq. (9) in [146], into Eq. (2.22):

$$\eta(X, Y, \tau) = A\mathrm{sech}^2\left[B\left(X + Y\tan\varphi - C\tau\right)\right], \tag{2.23}$$

where φ is the angle of incidence, A is the amplitude of the soliton, and B and C are coefficients to be determined via direct substitution. The KP soliton is then found to be

$$\eta(X, Y, \tau) = A\mathrm{sech}^2\left[\sqrt{\frac{3}{4}A}\left(X + Y\tan\varphi - C\tau\right)\right], \tag{2.24}$$

with $C = \frac{1}{2}A + \frac{1}{2}\tan^2\varphi$, $B = \sqrt{3A/4}$ and A the prescribed amplitude. Using Eq. (2.18) at leading order, i.e. $\eta = \Psi_X$, the solution for Ψ thus becomes

$$\Psi(X, Y, \tau) = \sqrt{\frac{4}{3}A}\left[\tanh\left(\sqrt{\frac{3}{4}A}\left(X + Y\tan\varphi - C\tau\right)\right) + 1\right]. \tag{2.25}$$

2.3 Comparison with modified Miles' theory and Kodama's exact solution

2.3.1 Introduction to Kodama's exact solution

Kodama *et al.* [84] have studied the reflection pattern for "symmetric V-shape initial waves consisting of two semi-infinite line solitons with the same amplitude", in a system of coordinates $(\tilde{X}, \tilde{Y}, \tilde{\tau})$ related to our system of coordinates (2.17) (X, Y, τ) via

$$\tilde{X} = \left(\frac{3}{\sqrt{2}}\right)^{1/3} X, \qquad \tilde{Y} = \left(\frac{3}{\sqrt{2}}\right)^{2/3} Y, \qquad \tilde{\eta} = \frac{1}{3}\left(\frac{3}{\sqrt{2}}\right)^{4/3}\eta \qquad \text{and} \quad \tilde{\tau} = \sqrt{2}\tau. \tag{2.26}$$

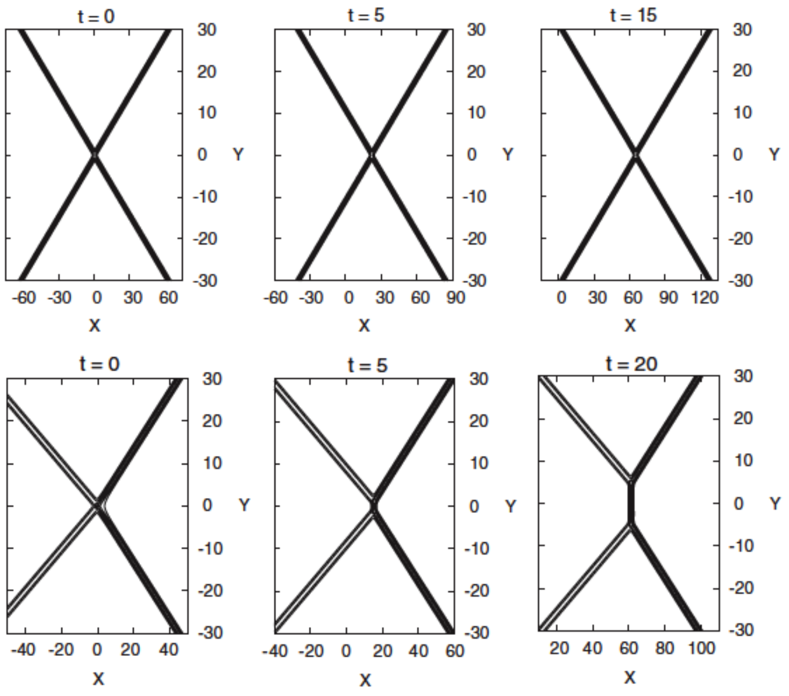


Figure 2.3: O-type and (3142)-type solitons as represented by Kodama *et al.* [84]. Top: evolution (from left to right) of the O-type soliton, consisting of two line solitons with different amplitudes and angles with respect to the y–axis. As it propagates, the shape of this soliton remains unchanged. Bottom: evolution (from left to right) of the (3142)-type soliton, consisting of two line solitons travelling in the x direction with different angles and amplitudes. As the soliton propagates, a new line soliton is created at the intersection of the two initial line solitons, leading to a stem wave. Figure obtained from [84]. ©IOP Publishing. Reproduced with permission. All rights reserved.

They solved the KP equation

$$[4\tilde{\eta}_{\tilde{\tau}} + 6\tilde{\eta}\tilde{\eta}_{\tilde{X}} + \tilde{\eta}_{\tilde{X}\tilde{X}\tilde{X}}]_{\tilde{X}} + 3\tilde{\eta}_{\tilde{Y}\tilde{Y}} = 0, \tag{2.27}$$

for which the surface deviation solution $\tilde{\eta}$ is given by

$$\tilde{\eta} = \tilde{A}\text{sech}^2\left[\sqrt{\frac{\tilde{A}}{2}}\left(\tilde{X} + \tilde{Y}\tan\tilde{\varphi} - \tilde{C}\tilde{\tau}\right)\right], \tag{2.28}$$

where $\tilde{A}$ is the amplitude of the soliton, $\tilde{\varphi}$ is the angle of incidence at the wall, and $\tilde{C}$ is a constant defined as $\tilde{C} \equiv \frac{1}{2}\tilde{A} + \frac{3}{4}\tan^2\tilde{\varphi}$. They showed that in this specific case, the transition from regular to Mach reflection occurs when

$$\tan\tilde{\varphi} = \sqrt{2\tilde{A}}. \tag{2.29}$$

Moreover, Kodama *et al.* [84] defined exactly the incident, reflected and stem solitons resulting from the interaction as an O-type soliton in the case where $\tan\tilde{\varphi} > \sqrt{2\tilde{A}}$, and a (3142)-type soliton in the case where $\tan\tilde{\varphi} < \sqrt{2\tilde{A}}$. The O-type soliton consists of two line solitons travelling in the x–direction, each having a specific amplitude and angle with respect to the y–axis (see Fig. 2.3). The (3142)-type soliton consists of two other line solitons, also travelling in the x–direction with their own amplitudes and angles with respect to the y–axis, but this soliton also has the property of being non-stationary, i.e. that while it propagates along the x–axis, a new line soliton is progressively created and grows parallel to the y–axis at the intersection of the two initial line solitons. In the case of both O-type and (3142)-type solitons, one of the line solitons can be associated with the incident solitary wave presented in the introduction, the second line solitons with the reflected wave (with a different amplitude and angle), and the intersection of the two line solitons as the stem wave, growing in length only when the angle of the incident wave is smaller than the critical angle (2.29). These two solitons are represented in Fig. 2.3, obtained from [84]. A comparison between these theoretical solitons and those obtained numerically from the V-shape initial soliton showed very good agreement, confirming that the incident, reflected and stem waves described by Miles are indeed asymptotically equivalent to the O-type and (3142)-type solitons, depending on the initial angles. In the case of a symmetric initial pattern, that is, for two initial line solitons of equal amplitude and angle of incidence, Kodama *et al.* [84] gave the expression of the maximal amplitude of the intersection wave as

$$a_{\max} = \begin{cases} \dfrac{1}{2}(\tan\tilde{\varphi} + \sqrt{2\tilde{A}})^2 & \text{for } \tan\tilde{\varphi} < \sqrt{2\tilde{A}}, \\ \dfrac{4\tilde{A}}{(1+\sqrt{1-\dfrac{2\tilde{A}}{\tan^2\tilde{\varphi}}})} & \text{for } \tan\tilde{\varphi} \geq \sqrt{2\tilde{A}}. \end{cases} \tag{2.30}$$

Since the condition $\tan\tilde{\varphi} = \sqrt{2\tilde{A}}$ is equivalent to Miles' condition $\kappa = 1$, we can define the interaction parameter corresponding to the KP Eq. (2.27) as

$$\tilde{\kappa} = \frac{\tan\tilde{\varphi}}{\sqrt{2\tilde{A}}}. \tag{2.31}$$

Substitution of the interaction parameter (2.31) into the amplification expectations (2.30) indeed yields Miles' predictions (2.4) for $\alpha_{\mathrm{w}} = a_{\mathrm{max}}/\tilde{A}$.

2.3.2 Application to the present Benney–Luke model

In section 2.2.3, the Benney–Luke model was reduced to the KP Eq. (2.22). This equation for the surface deviation η is slightly different from the one used by Kodama *et al.* [84] and introduced in Eq. (2.27). In order to compare our numerical solutions to Kodama *et al.*'s result [84], Eqs. (2.30)–(2.31), our KP Eq. (2.22) is (re)scaled using the coefficients introduced in Eq. (2.26), which yields Eq. (2.27) used by Kodama *et al.* [84]. Using the same transformations (2.26) in the KP soliton solution (2.28), we can obtain a solution for our KP Eq. (2.27) in terms of the original variables (X, Y, τ, η) introduced in Eq. (2.17), given by

$$\eta = 3\left(\frac{3}{\sqrt{2}}\right)^{-4/3}\tilde{A}\mathrm{sech}^2\left[\sqrt{\frac{\tilde{A}}{2}}\left(\left(\frac{3}{\sqrt{2}}\right)^{1/3}X - \tilde{C}\sqrt{2}\tau + \left(\frac{3}{\sqrt{2}}\right)^{2/3}Y\tan\tilde{\varphi}\right)\right]. \tag{2.32}$$

The connection between the above solution (2.32) and the previously presented solution (2.24) can be established by applying the following transformations in Eq. (2.32):

$$A = 3\left(\frac{3}{\sqrt{2}}\right)^{-4/3}\tilde{A}, \qquad C = \left(\frac{4}{3}\right)^{1/3}\tilde{C} \qquad \text{and} \qquad \tan\varphi = \left(\frac{3}{\sqrt{2}}\right)^{1/3}\tan\tilde{\varphi}, \tag{2.33}$$

with $C = \frac{1}{2}A + \frac{1}{2}\tan^2\varphi$, yielding the solution (2.24) derived in section 2.2.3. Therefore, applying scaling (2.33) to the critical condition (2.29) yields the critical condition for Eq. (2.22) and solution (2.24), given by

$$\tan\varphi = \sqrt{3A}. \tag{2.34}$$

We then apply scaling (2.17) to transform solution (2.24) for η back to the original Benney–Luke approximation (2.16) used in our simulations, in which case the asymptotic solutions for η and Φ

become

$$\eta(x,y,t) = A\text{sech}^2\left[\sqrt{\frac{3\epsilon}{4\mu}}A\left(x - x_0 + \sqrt{\epsilon}(y-y_0)\tan\varphi + (t-t_0)(1-C\epsilon)\right)\right], \tag{2.35a}$$

$$\begin{aligned}\Phi(x,y,t) = \sqrt{\frac{4\mu}{3\epsilon}A}\Bigg[\tanh\Bigg(\sqrt{\frac{3\epsilon}{4\mu}}A\Big(x - x_0 + \sqrt{\epsilon}(y-y_0)\tan\varphi \\ + (t-t_0)(1-C\epsilon)\Big)\Bigg) + 1\Bigg],\end{aligned} \tag{2.35b}$$

where the soliton has been localized around the position (x_0, y_0) at time $t = t_0$. Finally, by setting

$$a_\mathrm{i} = A, \qquad \tan\varphi_\mathrm{i} = \sqrt{\epsilon}\tan\varphi \qquad \text{and} \qquad \hat{C} = \frac{1}{2}a_\mathrm{i} + \frac{1}{2\epsilon}\tan^2\varphi_\mathrm{i}, \tag{2.36}$$

the solutions (2.35) of the Benney–Luke equations can be rewritten as

$$\eta(x,y,t) = a_\mathrm{i}\text{sech}^2\left[\sqrt{\frac{3\epsilon}{4\mu}}a_\mathrm{i}\left(x - x_0 + (y-y_0)\tan\varphi_\mathrm{i} + (t-t_0)(1-\hat{C}\epsilon)\right)\right], \tag{2.37a}$$

$$\begin{aligned}\Phi(x,y,t) =\sqrt{\frac{4\mu}{3\epsilon}a_i}\Bigg[\tanh\Bigg(\sqrt{\frac{3\epsilon}{4\mu}}a_i\Big(x - x_0 + (y-y_0)\tan\varphi_i \\ + (t-t_0)\left(1-\hat{C}\epsilon\right)\Big)\Bigg) + 1\Bigg].\end{aligned} \tag{2.37b}$$

This solution is used as an initial condition at time $t = 0$ in the simulations. Condition (2.34) defines the following relation between φ_i, a_i and ϵ in our Benney–Luke scaling, for Eq. (2.16):

$$\tan\varphi_\mathrm{i} = \sqrt{3\epsilon a_\mathrm{i}}. \tag{2.38}$$

This condition is equivalent to Miles' condition $\kappa = 1$ and therefore we can define our Benney–Luke interaction parameter as

$$\kappa_\mathrm{BL} = \frac{\tan\varphi_\mathrm{i}}{\sqrt{3\epsilon a_\mathrm{i}}}. \tag{2.39}$$

Note, however, that taking into account the remark from [83] about the quasi two-dimensionality of the KP limit, as explained in the introduction, the interaction parameter defined in Eq. (2.39) must be corrected to

$$\kappa_\mathrm{BL} = \frac{\tan\varphi_\mathrm{i}}{\cos\varphi_\mathrm{i}\sqrt{3\epsilon a_\mathrm{i}}} \tag{2.40}$$

in order to satisfy Miles' prediction (2.4). As shown in the potential-flow Eq. (2.10) for the Benney–Luke approximation, the small-amplitude parameter ϵ is defined as $\epsilon = a/H_0$. Therefore, in the specific case where $a_\mathrm{i} = 1$ and $\epsilon = a_\mathrm{KdV}/H_0$, the interaction parameter (2.8) is recovered. The diagram in Fig. 2.4 summarizes the equations and solutions derived thus far, in each scaling. In the next section, we explain how the Benney–Luke system of equations is discretized in both space and time in order to be solved numerically.

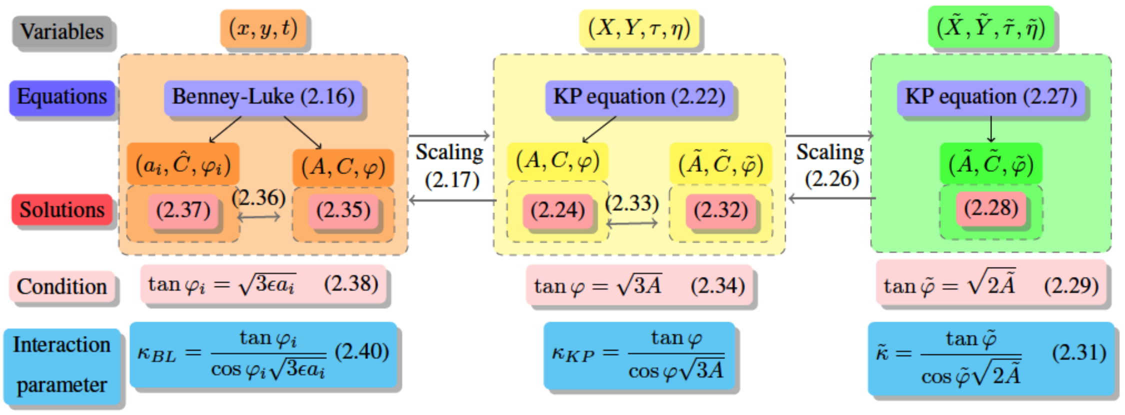

Variables
(x,y,t)
(X,Y,τ,η)
$(\tilde{X},\tilde{Y},\tilde{\tau},\tilde{\eta})$
Equations
Benney-Luke (2.16)
KP equation (2.22)
KP equation (2.27)
$(a_i,\hat{C},\varphi_i)$
(A,C,φ)
Scaling (2.17)
(A,C,φ)
$(\tilde{A},\tilde{C},\tilde{\varphi})$
Scaling (2.26)
$(\tilde{A},\tilde{C},\tilde{\varphi})$
Solutions
(2.37)
(2.36)
(2.35)
(2.24)
(2.33)
(2.32)
(2.28)
Condition
$\tan\varphi_i = \sqrt{3\epsilon a_i}$ (2.38)
$\tan\varphi = \sqrt{3A}$ (2.34)
$\tan\tilde{\varphi} = \sqrt{2\tilde{A}}$ (2.29)
Interaction parameter
$\kappa_{BL} = \frac{\tan\varphi_i}{\cos\varphi_i\sqrt{3\epsilon a_i}}$ (2.40)
$\kappa_{KP} = \frac{\tan\varphi}{\cos\varphi\sqrt{3A}}$
$\tilde{\kappa} = \frac{\tan\tilde{\varphi}}{\cos\tilde{\varphi}\sqrt{2\tilde{A}}}$ (2.31)

Figure 2.4: Schematic plan showing the link between the scaling of the three systems of equations involved in the derivation of the exact solution and critical condition for which Miles' and Kodama's predictions hold in the Benney–Luke approximation.

2.4 Numerical implementation

As a first step in the computational solution, the Benney–Luke model needs to be discretized in space and time, on a meshed domain. This section explains the methods used to discretize the domain and the equations.

2.4.1 Space discretization: finite-element method (FEM)

A continuous Galerkin finite-element method is used to discretize the solutions in space. The variables η and Φ are approximated by the finite-element expansions

$$\eta_h(x,y,t) = \eta_i(t)\varphi_\mathrm{i}(x,y) \qquad \text{and} \qquad \Phi_h(x,y,t) = \Phi_j(t)\varphi_j(x,y), \tag{2.41}$$

where the subscript h denotes the discretized form of the solutions with basis functions $\varphi_j(x,y)$, and $i,j \in [1,N]$ with $2N$ unknowns. The Einstein notation for the implicit summation of repeated indices is used. To limit restrictions on the finite-element expansions, the second-order derivative in the fourth term of the variational principle (2.15) is expressed through the auxiliary variable

$$q(x,y,t) = -\frac{2}{3}\Delta\Phi(x,y,t) \tag{2.42}$$

as suggested in [20], so that, in the variational principle (2.15), the term $\frac{\mu}{3}(\Delta\Phi)^2$ can be written as

$$\begin{aligned}\int_{\Omega_\mathrm{b}} \frac{\mu}{3}(\Delta\Phi)^2 \,\mathrm{d}\Omega_\mathrm{b} &= \int_{\Omega_\mathrm{b}} \mu\left(\frac{2}{3}(\Delta\Phi)^2 - \frac{1}{3}(\Delta\Phi)^2\right) \mathrm{d}\Omega_\mathrm{b}\\ &= \int_{\Omega_\mathrm{b}} \mu\left(-\frac{2}{3}\nabla\Delta\Phi\cdot\nabla\Phi - \frac{3}{4}(\frac{2}{3}\Delta\Phi)^2\right) \mathrm{d}\Omega_\mathrm{b}\\ &= \int_{\Omega_\mathrm{b}} \mu\left(\nabla q\cdot\nabla\Phi - \frac{3}{4}q^2\right) \mathrm{d}\Omega_\mathrm{b},\end{aligned} \tag{2.43}$$

which leads to the variational principle

$$\begin{aligned}0 = \delta\int_0^T \int_{\Omega_\mathrm{b}} \Big[&\eta\partial_t\Phi + \frac{\mu}{2}\nabla\eta\cdot\partial_t\nabla\Phi + \frac{1}{2}(1+\epsilon\eta)|\nabla\Phi|^2\\ &+\mu\left(\nabla q\cdot\nabla\Phi - \frac{3}{4}q^2\right) + \frac{1}{2}\eta^2\Big] \mathrm{d}\Omega_\mathrm{b}\,\mathrm{d}t.\end{aligned} \tag{2.44}$$

In keeping with Eq. (2.41), the second-order Galerkin expansion for q is now expressed as

$$q_h(x,y,t) = q_i(t)\varphi_\mathrm{i}(x,y). \tag{2.45}$$

Substituting expansions (2.41) and (2.45) into the variational principle (2.44) yields the space-discrete variational principle

$$\begin{aligned}0 = \delta\int_0^T \int_{\Omega_\mathrm{b}} \Big[&\varphi_j\eta_j\varphi_\mathrm{i}\dot{\Phi}_i + \frac{\mu}{2}\eta_j\dot{\Phi}_i\nabla\varphi_j\cdot\nabla\varphi_\mathrm{i} + \frac{1}{2}(1+\epsilon\varphi_j\eta_j)\Phi_i\Phi_l\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_l\\ &+\mu\left(q_i\Phi_j\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_j - \frac{3}{4}q_iq_j\varphi_\mathrm{i}\varphi_j\right) + \frac{1}{2}\varphi_\mathrm{i}\varphi_j\eta_i\eta_j\Big]\mathrm{d}\Omega_\mathrm{b}\,\mathrm{d}t,\end{aligned} \tag{2.46}$$

with $\dot{\Phi}_i$ the time derivative of Φ_i. The space-discrete variational principle (2.46) can also be written in matrix form

$$0 = \delta\int_0^T \left[\left(\mathbf{M}_{ij} + \frac{\mu}{2}\mathbf{A}_{ij}\right)\Phi_i\frac{\mathrm{d}\eta_j}{\mathrm{d}t} - H(\Phi_i,\eta_j)\right]\mathrm{d}t, \tag{2.47}$$

where $\mathbf{M}_{ij}$ and $\mathbf{A}_{ij}$ are the mass and stiffness matrices, respectively defined as

$$\mathbf{M}_{ij} = \int_{\Omega_{\mathrm{b}}} \varphi_{\mathrm{i}}\varphi_j \,\mathrm{d}x\,\mathrm{d}y \qquad \text{and} \qquad \mathbf{A}_{ij} = \int_{\Omega_{\mathrm{b}}} \nabla\varphi_{\mathrm{i}} \cdot \nabla\varphi_j \,\mathrm{d}x\,\mathrm{d}y, \tag{2.48}$$

and the Hamiltonian is

$$H(\phi_i, \eta_j) = \frac{1}{2}(\mathbf{A}_{ij} + \epsilon\mathbf{S}_{ijk}\eta_k)\Phi_i\Phi_j + \mu\left(\mathbf{A}_{ij}q_i\Phi_j - \frac{3}{4}\mathbf{M}_{ij}q_iq_j\right) + \frac{1}{2}\mathbf{M}_{ij}\eta_i\eta_j, \tag{2.49}$$

where

$$\mathbf{S}_{ijk} = \int_{\Omega_{\mathrm{b}}} \varphi_k\nabla\varphi_{\mathrm{i}} \cdot \nabla\varphi_j \,\mathrm{d}\Omega_{\mathrm{b}}. \tag{2.50}$$

Note that $\mathrm{d}H/\mathrm{d}t = 0$ due to skew symmetry. Rather than using this matrix form directly, we accommodate the spatial discretization using Firedrake [120, 7, 9, 36, 66], "an automated system for the portable solution of partial differential equations using the finite element method (FEM)". This automated system uses the finite-element method to solve partial differential equations, and requires specification of the following:

- the domain in which the equations are solved, and the kind of mesh to use (*e.g.* quadrilaterals, the spatial dimension);
- the order and type of polynomials used;
- the type of expansion for the unknowns (*e.g.* continuous Galerkin, Lagrange polynomials);
- the function space of the unknowns and test functions; and
- the weak formulations discretized in time.

In the present case the domain is defined as a horizontal channel ending in an oblique wall, and quadrilaterals are used for its discretization (see details in section 2.5.1). Here, we chose to use quadratic polynomials to expand Φ, q and η. The resulting weak formulations implemented in Firedrake in terms of Φ_h, q_h and η_h are obtained by taking the variations of Eq.(2.44) with end-point conditions $\delta\Phi(0) = \delta\Phi(T) = 0$:

$$\delta\eta_h : 0 = \int_0^T \int_{\Omega_b} \Big[\delta\eta_h \partial_t \Phi_h + \frac{\mu}{2}\nabla\delta\eta_h \cdot \nabla\partial_t\Phi_h + \eta_h\delta\eta_h + \frac{\epsilon}{2}\delta\eta_h \nabla\Phi_h \cdot \nabla\Phi_h\Big] \, \mathrm{d}\Omega_b \, \mathrm{d}t, \tag{2.51a}$$

$$\delta q_h : 0 = \int_0^T \int_{\Omega_b} \mu \left[\frac{3}{2} q_h \delta q_h - \nabla\delta q_h \cdot \nabla\Phi_h\right] \mathrm{d}\Omega_b \, \mathrm{d}t, \tag{2.51b}$$

$$\begin{aligned}\delta\Phi_h : 0 = \int_0^T \int_{\Omega_b} \Big[& -\partial_t\eta_h\delta\Phi_h - \frac{\mu}{2}\nabla\partial_t\eta_h \cdot \nabla\delta\Phi_h \\ & + (1+\epsilon\eta_h)\nabla\delta\Phi_h \cdot \nabla\Phi_h - \mu\nabla q_h \cdot \nabla\delta\Phi_h\Big] \mathrm{d}\Omega_b \, \mathrm{d}t.\end{aligned} \tag{2.51c}$$

The forms given in Eq. (2.51) are convenient since they highlight the unknowns Φ_h, q_h and η_h as well as the test functions $\delta\Phi_h$, δq_h and $\delta\eta_h$. The final step is to discretize the equations in time, with a second-order Störmer–Verlet scheme, as explained in the next section.

2.4.2 Time discretization: second-order Störmer–Verlet scheme

Gagarina *et al.* [52] have shown that, for a generic Hamiltonian system in the form

$$\delta\mathcal{L}(\mathbf{P}, \mathbf{Q}, t) = \delta \int_0^T \left(\mathbf{P}\frac{\mathrm{d}\mathbf{Q}}{\mathrm{d}t} - H(\mathbf{P}, \mathbf{Q})\right) \mathrm{d}t, \tag{2.52}$$

here with $\mathbf{P} = \{\Phi_i\}$ and $\mathbf{Q} = \{(\mathbf{M}_{ij} + \mu/2\mathbf{A}_{ij})\eta_j\}$, robust variational time integrators conserving the overall mass and energy can be formulated. To derive these time schemes, $\mathbf{P}$ and $\mathbf{Q}$ are discretized on each time interval $[t^n, t^{n+1}]$ as the approximated momentum $\mathbf{P}^\tau$ and coordinate $\mathbf{Q}^\tau$ and expanded with coefficients $\mathbf{P}^{\mathrm{m}}$ and $\mathbf{Q}^{\mathrm{m}}$ and linear continuous basis functions $\tilde{\psi}^m$ and ψ^m:

$$\mathbf{P}^\tau = \mathbf{P}^{\mathrm{m}}\tilde{\psi}^m(t) \qquad \text{and} \qquad \mathbf{Q}^\tau = \mathbf{Q}^{\mathrm{m}}\psi^m(t). \tag{2.53}$$

The linear basis functions $\tilde{\psi}^m$ and ψ^m are continuous within each time interval, but admit discontinuities at the interface between two time slots. Therefore, to discretize Eq. (2.52), the notion of jumps $[[.]]$ and averages $\{\{.\}\}_\alpha^\beta$ for a time-dependent function $d(t)$ must be introduced [52]:

$$[[d]]|_{t^n} = d^{n,-} - d^{n,+} \qquad \text{and} \qquad \{\{d\}\}_\alpha^\beta|_{t^n} = \alpha d^{n,-} + \beta d^{n,+}. \tag{2.54}$$

The coefficients α and β are real numbers defined such that $\alpha+\beta=1$ and $\alpha,\beta\geq 0$. The notation $d^{n,\pm}$ denotes the left and right traces of $d(t)$ at time t^n, that is,

$$d^{n,\pm} = \lim_{\epsilon\to 0} d(t^n \pm \epsilon). \tag{2.55}$$

Discretization of the variational principle (2.52) then yields [52]

$$\begin{aligned}\delta\mathcal{L}^{\tau}(\mathbf{P}^{\tau},\mathbf{Q}^{\tau},t) = \delta\Big[\sum_{n=0}^{N-1}\int_{t^n}^{t^{n+1}}\Big(\mathbf{P}^{\tau}\frac{\mathrm{d}\mathbf{Q}^{\tau}}{\mathrm{d}t} - H(\mathbf{Q}^{\tau},\mathbf{P}^{\tau})\Big)\mathrm{d}t \\ - \sum_{n=-1}^{N-1}[[\mathbf{Q}^{\tau}]]\{\{\mathbf{P}^{\tau}\}\}_{\alpha}^{\beta}|_{t^{n+1}}\Big],\end{aligned} \tag{2.56}$$

where N is the number of finite time intervals $[t^n,t^{n+1}]$ that divide the time domain $[0,T]$.

Among the time integrators derived in [52], the second-order Störmer–Verlet scheme was shown to be efficient and stable for the simulation of a soliton splash with the Benney–Luke–type model in [20]. While other choices can be made to increase the accuracy (with higher-order schemes, such as Runge-Kutta time scheme) or the computational speed (with lower-order schemes, such as the first-order symplectic-Euler scheme), we decide to apply the second-order Störmer–Velet scheme which balances the accuracy and computational cost of the numerical model. Gagarina *et al.* [52] showed that to obtain a second-order Störmer–Verlet scheme, $\mathbf{P}$ and $\mathbf{Q}$ must be discretized with mid-point and trapezoidal rules, respectively, that is,

$$\mathbf{Q}^{\tau} = \frac{t^{n+1}-t}{\Delta t}\mathbf{Q}^{n,+} + \frac{t-t^n}{\Delta t}\mathbf{Q}^{n+1,-}, \tag{2.57}$$

$$\mathbf{P}^{\tau} = \frac{2(t-t^n)}{\Delta t}\mathbf{P}^{n+1/2} + \frac{t^n+t^{n+1}-2t}{\Delta t}\mathbf{P}^{n,+}. \tag{2.58}$$

Substituting Eqs. (2.57)–(2.58) into the discretized variational principle (2.56) yields

$$\begin{aligned}\delta\mathcal{L}^{\tau}(\mathbf{P}^{\tau},\mathbf{Q}^{\tau},t) = \delta\left[\sum_{n=0}^{N-1}\Big(\mathbf{P}^{n+1/2}\left(\mathbf{Q}^{n+1,-}-\mathbf{Q}^{n,+}\right)\right. \\ -\frac{\Delta t}{2}\Big(H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n,+}) + H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n+1,-})\Big)\Big) \\ \left. - \sum_{n=-1}^{N-1}\left(\mathbf{Q}^{n+1,-}-\mathbf{Q}^{n+1,+}\right)\Big(2\alpha\mathbf{P}^{n+1/2}-\alpha\mathbf{P}^{n,+}+\beta\mathbf{P}^{n+1,+}\Big)\right].\end{aligned} \tag{2.59}$$

The variations of Eq. (2.59), when augmented by end-point conditions $\delta\mathbf{P}^{0,-} = \delta(2\mathbf{P}^{-1/2} - \mathbf{P}^{-1,+}) = 0$, $\delta\mathbf{Q}^{0,-} = 0$, $\delta\mathbf{P}^{N,+} = 0$ and $\delta\mathbf{Q}^{N,+} = 0$ to ensure that boundary conditions are satisfied, yield the following scheme:

$$\alpha\big(\mathbf{Q}^{n+1,-} - \mathbf{Q}^{n+1,+}\big) = \beta\big(\mathbf{Q}^{n,-} - \mathbf{Q}^{n,+}\big) \tag{2.60a}$$

$$\mathbf{P}^{n+1/2} = 2\alpha\mathbf{P}^{n+1/2} - \alpha\mathbf{P}^{n-1,+} + \beta\mathbf{P}^{n,+} - \frac{\Delta t}{2}\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n,+})}{\partial\mathbf{Q}^{n,+}}, \tag{2.60b}$$

$$\begin{aligned}(1-2\alpha)\mathbf{Q}^{n+1,-} + 2\alpha\mathbf{Q}^{n+1,+} = \mathbf{Q}^{n,+}& \\ + \frac{\Delta t}{2}\left(\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n,+})}{\partial\mathbf{P}^{n+1/2}} + \frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n+1,-})}{\partial\mathbf{P}^{n+1/2}}\right)&,\end{aligned} \tag{2.60c}$$

$$\beta\mathbf{P}^{n+1,+} = (1-2\alpha)\mathbf{P}^{n+1/2} + \alpha\mathbf{P}^{n,+} - \frac{\Delta t}{2}\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n+1,-})}{\partial\mathbf{Q}^{n+1,-}}. \tag{2.60d}$$

Setting $\mathbf{P}^n = \alpha\mathbf{P}^{n,+} + \beta\mathbf{P}^{n,-}$ with $\alpha = 0$ and $\beta = 1$ ensures stability of the numerical scheme [52]. Substituting these conditions into Eq. (2.60) yields the continuity condition $[[\mathbf{Q}]]_{t^n} = 0$ for $\mathbf{Q}$ in Eq. (2.60a), and the second-order Störmer–Verlet scheme is recovered,

$$\mathbf{P}^{n+1/2} = \mathbf{P}^n - \frac{\Delta t}{2}\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^n)}{\partial\mathbf{Q}^n}, \tag{2.61a}$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \frac{\Delta t}{2}\left(\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^n)}{\partial\mathbf{P}^{n+1/2}} + \frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n+1})}{\partial\mathbf{P}^{n+1/2}}\right), \tag{2.61b}$$

$$\mathbf{P}^{n+1} = \mathbf{P}^{n+1/2} - \frac{\Delta t}{2}\frac{\partial H(\mathbf{P}^{n+1/2},\mathbf{Q}^{n+1})}{\partial\mathbf{Q}^{n+1}}, \tag{2.61c}$$

with the stability condition

$$|\omega\Delta t| \leq 2, \tag{2.62}$$

where ω is the (maximum) frequency of the discrete waves. Demonstration of stability condition (2.62) and dermination of ω are given in Chapter 3, section 3.4.4. Setting the vectors $\mathbf{P} = \{\Phi_i\}$ and $\mathbf{Q} = \left\{\left(\mathbf{M}_{ij} + \frac{\mu}{2}\mathbf{A}_{ij}\right)\eta_j\right\}$, the variational principle (2.47) for Benney–Luke equations can therefore be discretized as in Eq. (2.61), leading to Eq. (A.7) in Appendix A.1. Since the space discretization is performed internally within Firedrake, the weak formulations (A.7) can be implemented with the full form of the variables Φ_h, q_h and η_h and test functions $\delta\Phi_h$, δq_h and $\delta\eta_h$

yielding the time discretization of Eq. (2.51), namely

$$\begin{aligned}0 = \int_{\Omega_b} &\left(\Phi_h^{n+1/2} - \Phi_h^n\right)\delta\eta_h + \frac{\mu}{2}\nabla\delta\eta_h \cdot \nabla\left(\Phi_h^{n+1/2} - \Phi_h^n\right)\\ &+ \frac{\Delta t}{2}\left[\eta_h^n\delta\eta_h + \frac{\epsilon}{2}\delta\eta_h\nabla\Phi_h^{n+1/2}\cdot\nabla\Phi_h^{n+1/2}\right]\mathrm{d}\Omega_b,\end{aligned} \tag{2.63a}$$

$$0 = \int_{\Omega_b}\left(q_h^{n+1/2}\delta q_h - \frac{2}{3}\nabla\delta q_h\cdot\nabla\Phi_h^{n+1/2}\right)\mathrm{d}\Omega_b, \tag{2.63b}$$

$$\begin{aligned}0 = \int_{\Omega_b} &\left(\eta_h^{n+1} - \eta_h^n\right)\delta\Phi_h + \frac{\mu}{2}\nabla\delta\Phi_h\cdot\nabla\left(\eta_h^{n+1} - \eta_h^n\right)\\ &- \frac{\Delta t}{2}\Big[\left((1+\epsilon\eta_h^n)\nabla\delta\Phi_h\cdot\nabla\Phi_h^{n+1/2} - \mu\nabla q_h^{n+1/2}\cdot\nabla\delta\Phi_h\right)\\ &+ \left((1+\epsilon\eta_h^{n+1})\nabla\delta\Phi_h\cdot\nabla\Phi_h^{n+1/2} - \mu\nabla q_h^{n+1/2}\cdot\nabla\delta\Phi_h\right)\Big]\mathrm{d}\Omega_b,\end{aligned} \tag{2.63c}$$

$$\begin{aligned}0 = \int_{\Omega_b} &\left(\Phi_h^{n+1} - \Phi_h^{n+1/2}\right)\delta\eta_h + \frac{\mu}{2}\nabla\delta\eta_h\cdot\nabla\left(\Phi_h^{n+1} - \Phi_h^{n+1/2}\right)\\ &+ \frac{\Delta t}{2}\left[\eta_h^{n+1}\delta\eta_h + \frac{\epsilon}{2}\delta\eta_h\nabla\Phi_h^{n+1/2}\cdot\nabla\Phi_h^{n+1/2}\right]\mathrm{d}\Omega_b.\end{aligned} \tag{2.63d}$$

Time-step Eqs. (2.63a), (2.63b) and (2.63c) are implicit, while Eq. (2.63d) is explicit. Although the equations are nonlinear, the step Eqs. (2.63b), (2.63c) and (2.63d) are linear with respect to the unknowns, $q_h^{n+1/2}$, η_h^{n+1} and Φ_h^{n+1}, respectively. Therefore, linear solvers are used to solve the three weak formulations (2.63b, c, d), which reduces the computational cost. The implementation of such linear and nonlinear solvers is straightforward in Firedrake, since functions that solve weak formulations for specific unknown and test functions already exist [120, 9, 7, 66, 36]. Details on the implementation are given in Chapter 6.

2.5 Numerical results

In this section, the domain is specified and discretized in order to evaluate Φ and η numerically. The numerical evolution of the stem-wave amplitude is compared to the predictions from our modified Miles' theory Eqs. (2.4) and (2.40). Finally, the angles of propagation of the reflected and stem waves are measured and compared to the values predicted by theory.

2.5.1 Definition of the domain

Orientation of the channel

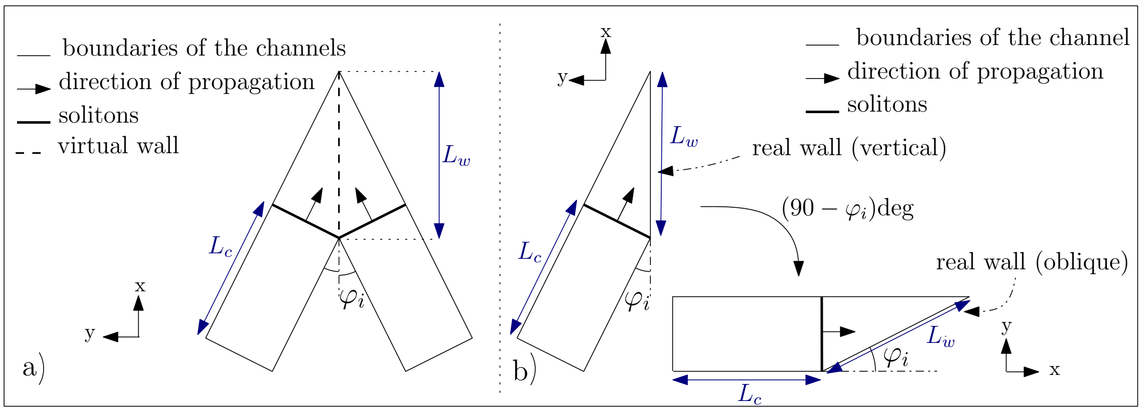


Figure 2.5: Definition of the domains in the two cases described in the text: **(a)** intersection of two channels, with two obliquely incident solitons interacting at a virtual wall, and **(b)** half of the domain with a soliton propagating in one channel and colliding with an oblique wall. This wall is in the x–direction (in which case the soliton has a two-dimensional propagation of direction) or oblique, in which case the incident soliton propagates in the one-dimensional direction (x).

The interaction of two solitary waves can be modelled using either two obliquely intersecting channels, with incident solitons propagating along each channel (see scheme a) in Fig. 2.5), or from the reflection of a soliton at a wall with the no-normal flow condition at the wall (see scheme b) in Fig. 2.5). While the case a) is more relevant to the theme of this chapter, we choose to model case b) to reduce the size of the domain by half and thus to reduce the computational cost. Since cases a) and b) are mathematically equivalent, the results and conclusions obtained with half of the domain will also be valid for the intersection of two oblique channels.

The domain is described by the length of the wall L_w, the length of the channel L_c, and the angle of incidence φ_i. The channel should be long enough, compared to the wavelength of the incident wave, in order for the boundaries to be far enough from the initial soliton to be considered infinitely distant. From Eq. (2.35), the width of the initial soliton depends on $\sqrt{3\epsilon/4\mu}$; for every simulation, μ is set to 0.02 while ϵ varies from $\epsilon = 0.20$ to $\epsilon = 0.12$, thus yielding soliton width from 2.5 to 4. We set $L_{\mathrm{c}} = 5$ to leave enough space between the extremities of the soliton and the boundary of the channel for every case, for the extremities to be considered infinitely distant from the soliton boundaries. To allow the stem wave to grow and reach its maximal amplitude, the wall also needs to be long compared to the wavelength. This constraint was a limit in previous numerical and experimental studies [133, 89] since it requires robust and stable numerical schemes and large wave basins. We set the wall length to $200 \leq L_{\mathrm{w}} \leq 600$ depending on the value of ϵ, that is, more than 100 times the incident-wave width. When considering half of the domain represented in Fig. 2.5b, we chose to set the wall in the x–direction, in which case the initial soliton must propagate in an oblique direction and is therefore equivalent to a KP soliton, as defined in Eq. (2.37). Alternatively, we can let the initial soliton propagate in the x–direction, in which case the wall is oblique and the expression of the KP-type soliton (2.37) can be simplified to a KdV-type soliton propagating in the x–direction, as [40]

$$\eta(x,y,t) = a_i \mathrm{sech}^2\left[\sqrt{\frac{3\epsilon}{4\mu}a_i}\Big(x - x_0 + (t-t_0)\left(1-\hat{C}\epsilon\right)\Big)\right], \tag{2.64a}$$

$$\Phi(x,y,t) = \sqrt{\frac{4\mu}{3\epsilon}a_i}\left[\tanh\left(\sqrt{\frac{3\epsilon}{4\mu}a_i}\Big(x - x_0 + (t-t_0)\left(1-\hat{C}\epsilon\right)\Big)\right) + 1\right]. \tag{2.64b}$$

The behaviour of the incident and stem waves in the cases of an oblique incident soliton (2.37) and a soliton propagating in the x–direction only (2.64) are compared in Fig. 2.6. The initial solitons have amplitude $a_{\mathrm{i}} = 1.0$, small-amplitude parameter $\epsilon = 0.14$ and small-dispersion parameter $\mu = 0.02$. The angle between the direction of propagation of the solitons and the wall is $\varphi_{\mathrm{i}} = \pi/6$ in both cases. The dashed lines represent the evolution of the interpolated amplitude of incident solitons with time. While the initial amplitude was $a_{\mathrm{i}} = 1.0$ in both cases, we observe that both amplitudes first increase before decreasing to an asymptotic value slightly smaller than 1.0 ($a_{\mathrm{i}} = 0.93$). This behaviour is not expected for solitons, which should keep a permanent shape. However, we solve here the Benney–Luke equations for which the KP soliton is only an asymptotic (and hence not exact) solution because we recall that the transformation (2.17) from the Benney–Luke model to the KP theory is not exact since it requires a truncation to $O(\epsilon^2)$. In the numerical simulations represented in Fig. 2.6, $\epsilon = 0.14$, so the condition $\epsilon \ll O(1)$ is respected only asymptotically; this is a possible explanation of the observed variation in amplitude. Figure 2.6 shows that the incident KP and KdV-type solitons Eqs. (2.37) and (2.64) converge, and that both

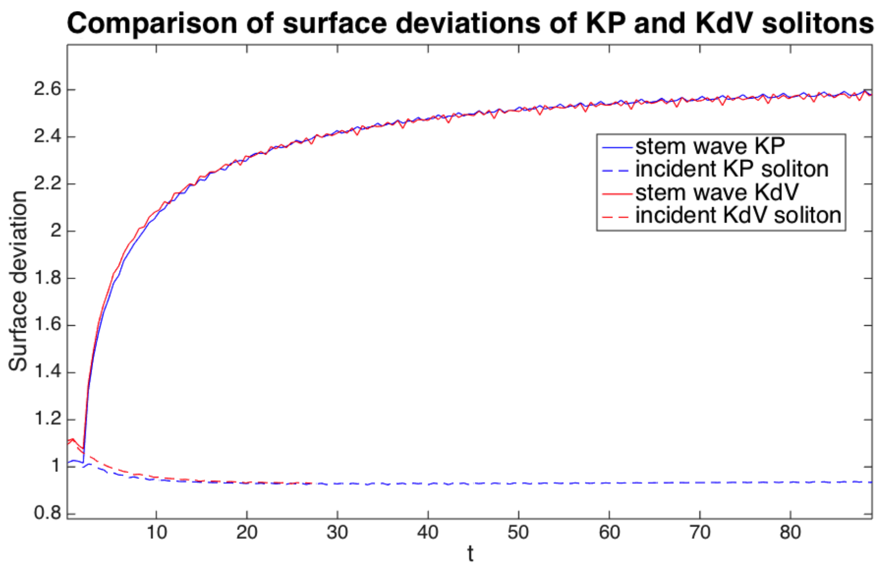


Figure 2.6: Soliton surface deviations obtained for an initial amplitude $a_\mathrm{i} = 1.0$ and angle $\varphi_\mathrm{i} = \pi/6$ rad. Blue: behaviour of the incident (dashed line) and stem (full line) waves when the incident soliton propagates in an oblique direction; red: behaviour of the incident (dashed line) and stem (full line) waves when the incident soliton propagates in one direction. The dashed lines essentially coincide after $t > 30$.

do so to the same surface deviation, $a_\mathrm{i} = 0.93$. This same limit shows that the approximation error from Benney–Luke to the KP soliton is asymptotically the same as from Benney–Luke to KdV. The stem-wave amplitudes (solid lines in Fig. 2.6) resulting from the interaction of the KP-type (2.37) and KdV-type (2.64) initial solitons with the wall are both amplified at the same speed and with the same amplification factor, which confirms that the KP-type and KdV-type initial solitons Eqs. (2.37) and (2.64) give the same results. The small variations in the curves are due to the mesh resolution which is not fine enough to resolve a regular amplitude. However, the computed approximation is sufficiently accurate to provide an estimate of the asymptotic amplitude of the stem wave. Since we have demonstrated that the two types of initial solitons Eqs. (2.37) and (2.64) evolve similarly to give the same results, subsequent simulations will be conducted using only a unidirectional soliton, as defined by Eq. (2.64), which is a solution of both the KP and KdV equations.

Mesh

In order to evaluate Φ and η at any position in the channel, the domain is discretized using quadrilaterals. This is done using the Gmsh mesh generator [56]. Since the domain is large, we define a heterogeneous mesh with areas of higher refinement along the wall, where the solution needs to be more accurate. Moreover, the end of the domain is truncated with a blunt wall instead of the sharp angle, to avoid boundary quadrilaterals having internal angles that are too acute. The final domain comprising different mesh refinements is presented in Fig. 2.7, in which the insets show the aforementioned refined mesh and right-hand boundary quadrilateral elements.

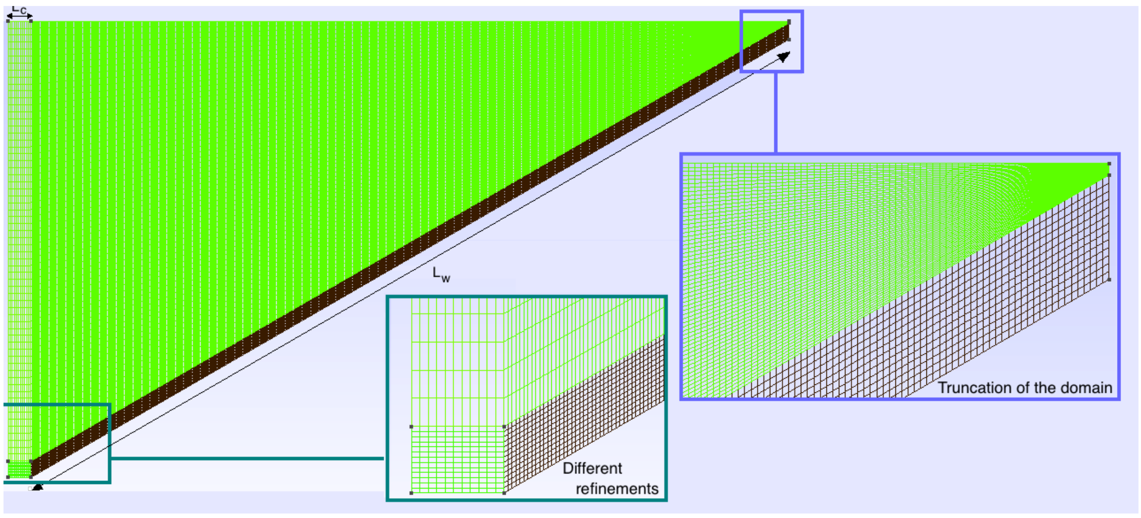


Figure 2.7: Domain discretization using quadrilaterals in Gmsh. In order to reduce computational requirements, mesh refinement is restricted to only the region adjacent to the wall.

2.5.2 Amplification of the stem wave

The numerical amplification of the stem wave is compared with the predictions of modified Miles' theory applied to our Benney–Luke model Eqs. (2.4) and (2.40), namely

$$\alpha_{\mathrm{w}} = \begin{cases} \dfrac{4}{1+\sqrt{1-\kappa^{-2}}}, & \text{for } \kappa \geq 1, \\ (1+\kappa)^2, & \text{for } \kappa < 1, \end{cases} \tag{2.4}$$

$$\text{with } \kappa = \frac{\tan\varphi_{\mathrm{i}}}{\cos\varphi_{\mathrm{i}}\sqrt{3\epsilon a_{\mathrm{i}}}}. \tag{2.40}$$

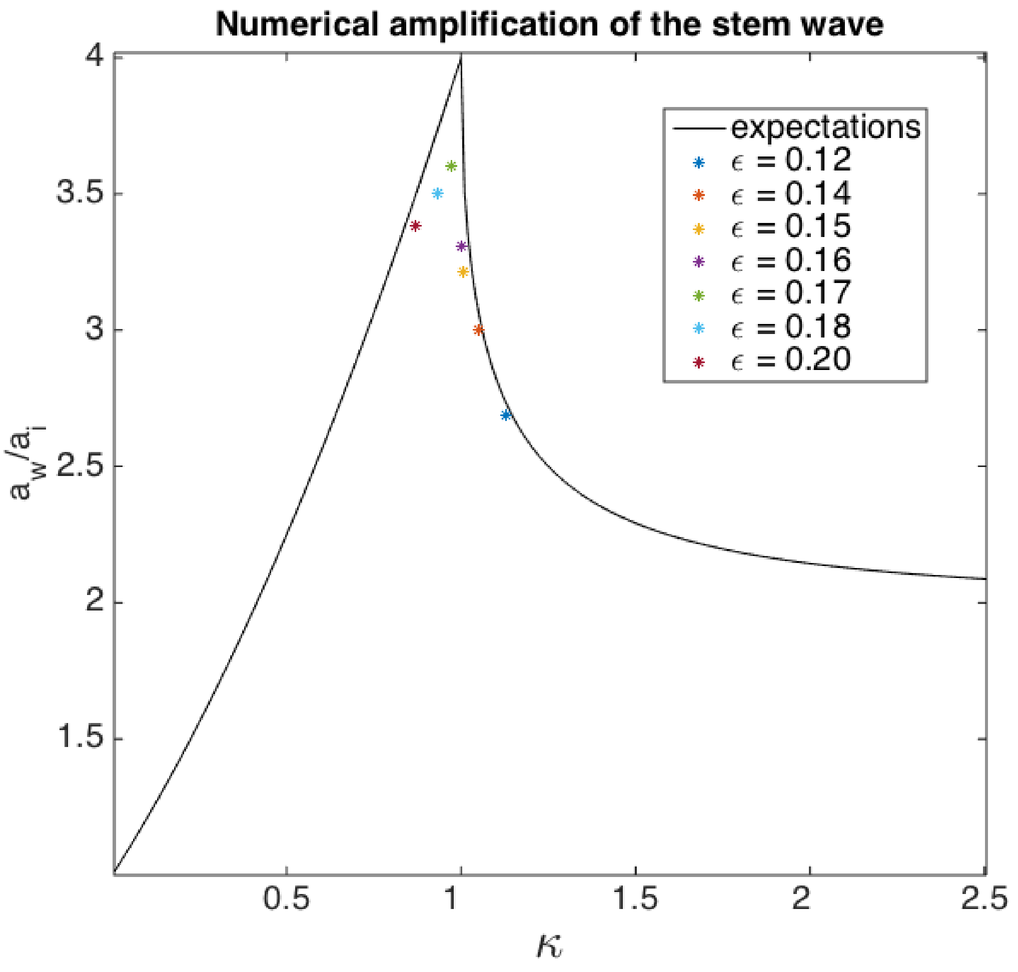


Figure 2.8: Comparison between the expected amplification (solid line) from Miles (2.4) and our numerical results (symbols) for different values of the interaction parameter κ, namely $\kappa \approx 1.1265$ ($\epsilon = 0.12$), $\kappa \approx 1.0526$ ($\epsilon = 0.14$), $\kappa \approx 1.0077$ ($\epsilon = 0.15$), $\kappa \approx 0.9989$ ($\epsilon = 0.16$), $\kappa \approx 0.9733$ ($\epsilon = 0.17$), $\kappa \approx 0.9345$ ($\epsilon = 0.18$), and $\kappa \approx 0.8692$ ($\epsilon = 0.20$).

The interaction parameter defined in Eq. (2.40) depends on three parameters: the scaled amplitude of the incident soliton a_{i}, its angle of incidence φ_{i}, and the small-amplitude parameter ϵ. From Miles' theory, a change in these parameters will modify the behaviour of the reflected and stem waves. Figure 2.8 shows a comparison between predictions Eqs. (2.4) and (2.40) and numerical simulations for the maximal amplification of the stem wave. The amplitude and angle of incidence of the initial soliton are the same for each of the simulations, with values $a_i = 1.0$ for the amplitude and $\varphi_i = 30$ degrees for the angle of incidence. Only the small-amplitude parameter ϵ changes in the different cases, taking values from 0.12 to 0.20, which leads to different interaction parameters and thus different evolutions of the stem and reflected waves. Variation of ϵ is an alternative choice to the one made in the work of Ablowitz and Curtis [1], where, for a specific ϵ, they compute simulations with varying amplitude and angle of incidence; this choice enabled them to show that the small-amplitude parameter ϵ has only a weak impact on the amplification of the stem wave for $\kappa < 1$ but limits the amplification, with a decrease of $O(\epsilon)$ close to the resonant case $\kappa = 1$, leading, for example, to a maximal wave amplification of 3.9 when $\epsilon = 0.1$. Despite this asymptotic limitation in the wave amplification, the purpose of the present simulations is to model wave amplification in various sea states, with various depths of water and characteristic wave heights, and we do so by using different values of ϵ, recalling that the small-amplitude parameter ϵ is the quotient between the characteristic wave height and the water depth. Modelling various sea states will allow the maritime industry to test wave impact on a wider range of structures, since different structures are used in different sea states. Moreover, the incident-wave amplitude varies slightly when propagating along the basin, which has a high impact on the predictions. Indeed, a small change of order $O(10^{-2})$ in the incident-wave amplitude implies a change of order $O(10^{-2})$ in the interaction parameter, which can lead to a prediction variation of up to $O(10^{-1})$ near the transition case $\kappa \approx 1$, in which the expected amplification varies dramatically. The amplification a_w/a_i is also affected by a change in the incident amplitude a_i. It is therefore necessary to use the accurate value for the incident amplitude. In performing the computations required for Fig. 2.8, we defined the maximal amplification as follows: when the stem wave reaches its maximal amplitude a_{w_max}, we measure the amplitude of the incident wave a_i at the same x–position. The new incident amplitude a_i is used to adjust the interaction parameter and to compute the amplification of the stem wave $\alpha_\text{w} = a_{w_\text{max}}/a_\text{i}$. The grid refinement is 0.25×0.25 in the finest area (*e.g.* at the wall) and 0.4×1.5 elsewhere. The numerics follow the theoretical curve, but a slight difference between the present results and those expected from modified Miles is noticeable. As alluded to beforehand, we assume that this is due to the fact that the soliton used as an incident wave is an asymptotic but not an exact solution of the Benney–Luke equations. The scaling from Benney–Luke to KP is not exact but asymptotic, with a truncation at second order, which leads to a slight difference in the final wave amplification. This observation agrees with the conclusions of Ablowitz and Curtis [1] on the asymptotic amplification of the stem wave in the case of the Benney–Luke model. The shift is probably also increased by the mesh resolution, which could be optimized to get a better estimate of the incident-wave amplitude in order to limit the error caused by its approximation. New simulations with higher mesh resolution are expected to verify the current results. However, the present Benney–Luke model still predicts the evolution of the stem-wave amplitude very well, enabling it to reach up to 3.6 times the initial amplitude. The stem-wave maximal amplification is reached for $\kappa = 0.9733$, marginally smaller than the $\kappa = 1.0$ predicted by Miles. While the model from Kodama *et al.* [84] is expected to predict the evolution of the stem wave based on the KP equation perfectly, they were unable to reach more than 3.2 times the initial amplitude in their numerical simulations.

2.5.3 Angle of the stem and reflected waves

Miles' theory also predicts different directions of propagation of the stem and reflected waves in the cases of regular and Mach reflections. While in the first case, characterized by $\kappa \geq 1$, the angle of the reflected wave φ_r is expected to be equal to that of the incident soliton φ_i, it should become larger than φ_i in the case of Mach reflection, i.e. when $\kappa < 1$:

$$\begin{cases} \varphi_r = \varphi_i & \text{for} \quad \kappa \geq 1, \\ \varphi_r > \varphi_i & \text{for} \quad \kappa < 1. \end{cases} \tag{2.65}$$

Moreover, in the case of regular reflection, the stem wave is expected to propagate along the wall with a constant length, while for Mach reflection, its length should increase linearly, making a non-zero angle φ_w with the wall:

$$\begin{cases} \varphi_w = 0 & \text{for} \quad \kappa \geq 1, \\ \varphi_w > 0 & \text{for} \quad \kappa < 1. \end{cases} \tag{2.66}$$

Predictions Eqs. (2.65) and (2.66) are checked numerically next.

Regular reflection

Figure 2.9 shows numerical results and predictions for the case where $\kappa = 1.12 \geq 1$. The wall makes an angle of 30 degree with the direction of propagation of the initial solitary wave, hence

$\varphi_{\rm i} = 30$ degree. In the bottom-right plot of Fig. 2.9, there is an angle of 60 degree between the reflected and stem waves, which means that the angle $\varphi_{\rm r}$ between the reflected wave and the line perpendicular to the wall is equal to 30 degree, that is, equal to $\varphi_{\rm i}$. This observation holds at any time, and therefore the expectations (2.65) for the reflected waves are satisfied in the case of regular reflection. The stem wave propagates along the wall without increasing in length, and therefore no angle can be measured between the stem wave and the wall, i.e. $\varphi_{\rm w} = 0$, as predicted in Eq. (2.66) for regular reflection. These results, together with Fig. 2.8 for the amplification of the stem wave, are consistent with modified Miles' theory in the case $\kappa \geq 1$ for both the reflected and stem waves.

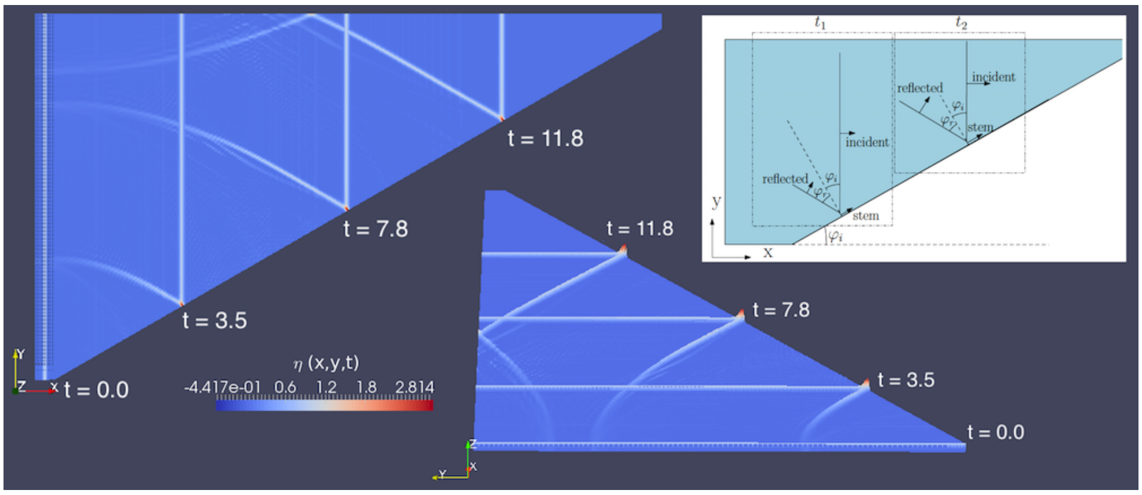


Figure 2.9: Numerical results and predictions for the reflected and stem waves in the case of regular reflection, i.e. $\kappa > 1$. Top left: top view of the numerical evolution of the incident, reflected and stem waves. Top right: schematic plan view of the expected evolution of the stem and reflected waves at two different times t_1 and t_2 with $t_1 < t_2$. The stem wave should propagate along the wall with constant length. The angle $\varphi_{\rm r}$ of the reflected wave is expected to be constant and equal to the incident-wave angle $\varphi_{\rm i}$. Bottom centre: side view of the time evolution of the incident, reflected and stem waves, highlighting the amplification of the stem-wave amplitude compared to the initial solitary-wave height.

Mach reflection

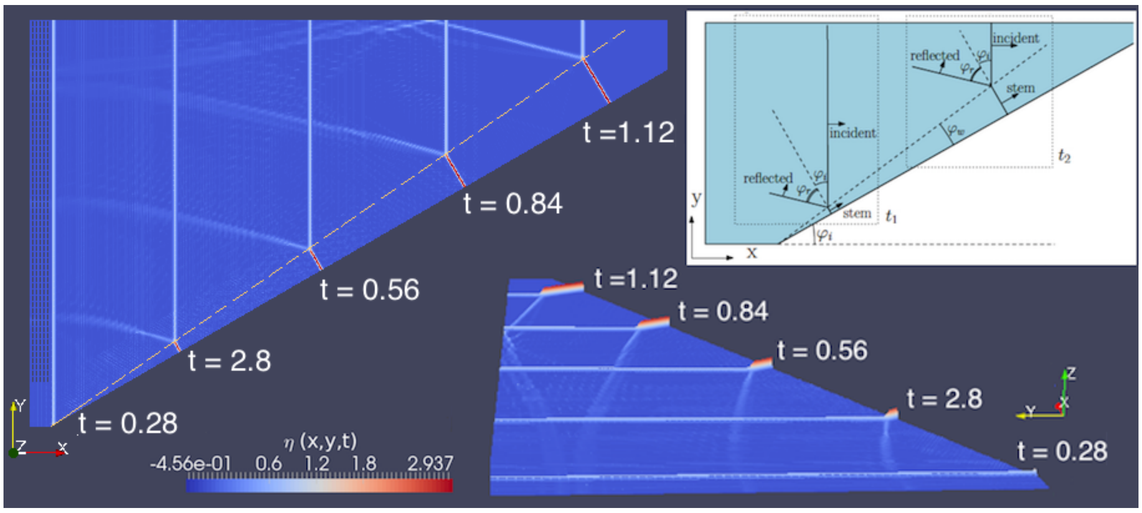


Figure 2.10: Numerical results and predictions for the reflected and stem waves in the case of Mach reflection, i.e. $\kappa < 1$. Top left: schematic plan view of the numerical evolution of the incident, reflected and stem waves. Top right: top-view scheme of the predicted evolution of the stem and reflected waves at two different times t_1 and t_2 with $t_1 < t_2$. The stem wave should grow linearly in length, leading to an angle $\varphi_\mathrm{w} > 0$ with the wall. The angle φ_r of the reflected wave is expected to be constant and larger than the incident-wave angle φ_i. Bottom centre: side view of the time evolution of the incident, reflected and stem waves, highlighting the amplification of the stem-wave amplitude compared to the initial solitary-wave height.

Figure 2.10 shows numerical results and schematic expectations for the propagation of the reflected and stem wave for $\kappa = 0.58 < 1$. In the bottom-right sub-figure, the angle between the incident and reflected waves can be measured, as represented in the top-right sub-figure, in order to check that φ_r is larger than φ_i. The total angle $\varphi_\mathrm{r} + \varphi_\mathrm{i}$ measures 70 degree, with the initial incident angle set to $\varphi_\mathrm{i} = 30$ degree. Therefore, φ_r is 40 degree, which is indeed larger than φ_i, thereby agreeing with our predictions. The top-right sub-figure of Fig. 2.10 also shows that the stem-wave length should increase linearly to form an angle φ_w with the wall. In the bottom-right figure, a top view of the numerical results at different times from $t = 0.28$ to $t = 1.12$ highlights the increase in the stem wave's length as it propagates along the wall. The dashed orange line that connects the solutions in the left subfigure confirms that the wavelength increases linearly. Therefore, predictions (2.66) are also verified in the case of Mach reflection.

2.6 Conclusions and discussions

The present model (2.16), together with the new scaled interaction parameter (2.40), shows good agreement with the predictions of Miles regarding the amplification of the stem wave and the angles of the reflected and stem waves. Two different regimes can be observed in the numerical results, with different behaviours of the waves in the case of Mach and regular reflections, which confirms the conclusions obtained by Ablowitz and Curtis [1] regarding the ability of the Benney–Luke model to predict reflection of obliquely incident solitary waves. Due to limited computational resources, the resolution used in our simulations does not allow determination of the exact value of the interaction parameter at the transition from Mach to regular reflection, but currently the maximal amplification is reached at $\kappa = 0.9733$, which is very close to the predicted maximal amplification at $\kappa = 1.0$. The maximal amplification obtained herein is $\alpha_\text{w} = 3.6$, which is higher than the amplifications obtained with most previous models and experiments [84, 89, 133, 50], but still slightly lower than the expected 3.9 amplification from Ablowitz and Curtis [1]. This agrees with the conclusion of [1] concerning the impact of ϵ on the amplification near $\kappa = 1$. While they obtained the maximal amplification $\alpha_\text{w} = 3.9$ for $\epsilon = 0.10$, our amplification $\alpha_\text{w} = 3.6$ is obtained for $\epsilon = 0.17$, which is larger than 0.1 and thus leads to a larger difference with Miles' prediction of $\alpha_\text{w} \approx 4$. Moreover, thanks to the robust scheme used to derive and discretize our equations, which ensures stable simulations over the large domain despite the different length scales involved, our model is the first model able to describe numerically the dynamic development of the stem wave up to such high amplitudes. Previous studies [84, 89, 133, 50] were not able to attain such high amplifications because of numerical limitations such as insufficient computational resources. Ablowitz and Curtis [1] obtained the highest numerical amplification $\alpha_\text{w} = 3.9$ by considering the final state, initialized asymptotically using the KP two-line solution. This last approach gives an accurate understanding of the asymptotic maximal amplification of the stem wave with the Benney–Luke model, but does not describe the development of the stem wave along the wall. The description and understanding of the wave propagation along the wall is however fundamental for application purposes. The present results, although currently limited by computational resources, allow us to consider the relevance of obliquely interacting solitary waves in maritime engineering. More advanced simulations should enable determination of the value of κ at the transition from Mach to regular reflection and to reach higher amplification of the stem wave.

One may wonder how likely it is that solitary waves would undergo reflection in an open ocean. Interaction of obliquely incident waves on the sides of a ship leads to an increasing wave amplitude, sometimes reaching the deck. This phenomenon is called "green water" and has been studied experimentally and numerically by the Maritime Research Institute Netherlands (MARIN) to limit the damage caused by waves on ships [23]. When the incident wave interacts with a ship moving downwind, the effective ship length increases, leaving more time for the stem wave to develop to its maximum amplitude. Peterson *et al.* [118] also studied the formation of extreme waves in shallow water and explained under which conditions they are likely to occur and threaten ships. Kalogirou and Bokhove [77] developed numerical models of waves impacting buoys and ships. An extension of our oblique-wave interaction simulations to wave interactions with ships will be an interesting extension of our present work.

The present model can also be used to predict the impact of extreme (i.e. freak or rogue) waves on structures. Indeed, when the stem wave reaches more than twice the amplitude of the incident wave, it can be viewed as a freak wave since it has similar properties in terms of nonlinearity, dispersivity and high amplitude. Table 2.1 shows the distance required by the stem wave to reach more than twice the incident-wave amplitude in several cases parameterized by different values of ϵ. For each value of the small-amplitude parameter ϵ, the numerical (dimensionless) distance L_{n} required to reach at least twice the amplitude of the initial wave has been measured from the simulations. Then, the definition of the small-amplitude parameter $\epsilon = a_0/H_0$ and the choice of a sea state with characteristic wave height $a_0 = 3$ m enables computation of the corresponding water depth H_0. The physical distance L_r required by the wave to propagate up to twice the characteristic wave height can then be obtained from scaling (2.12), using $L_r = L_{\text{n}} \times H_0/\sqrt{\mu}$.

The value of the small-dispersion parameter μ is set to 0.02 as in the results section, thus leading to various wavelength λ_0 that can be obtained from the definition of the small-dispersion parameter $\mu = (H_0/\lambda_0)^2$. In a wave tank where waves can be generated in several directions, the angle of propagation and initial profile of two solitary waves can be defined from the asymptotically exact solution (2.37) of our model (2.16), so that their interaction will lead to a stem wave. The evolution of the stem wave can be predicted from the present model, so an offshore structure such as a scaled ship or a wind turbine can be placed at a position where the stem wave will reach more than twice the initial amplitude of the solitary waves. For instance, a scaling of $1/10$ between the values of H_0, L_r and λ_0 in Table 2.1 and experiments leads to achievable incident-wave amplitudes and distances of propagation in MARIN's shallow-water basin. From the amplitude of the stem wave at a given position, the impact of the wave on structures can be estimated and the predictions yielded by the model tests can be validated. The model can help the maritime industry to design safer offshore structures that can resist extreme-wave impacts.

	ϵ						
	0.12	0.14	0.15	0.16	0.17	0.18	0.20
Numerical distance L_n	5.8	5.5	5.5	7.8	7.7	8.0	8.0
Water depth H_0 (m)	25.00	21.43	20.00	18.75	17.65	16.67	15.00
Real distance L_r (m)	1025	833	778	1028	965	940	846
Wavelength λ_0 (m)	176.78	151.52	141.42	132.58	124.78	117.85	106.07

Table 2.1: Prediction of the minimal distance needed by the stem wave to reach at least twice its initial amplitude in a sea state with characteristic wave height $a_0 = 3\,\text{m}$. The dispersion parameter μ is set to 0.02, while the small-amplitude parameter ϵ varies from 0.12 to 0.20, leading to different wave evolutions. The numerical distance needed to reach more than twice the incident-wave amplitude is measured from the numerical simulations. The corresponding water depth, real distance of propagation and wavelength are computed from the definition of ϵ, μ and scaling (2.12). These values are approximate.

Finally, the present work can also be used as a starting point for the modelling of the interaction of three obliquely incident line solitons, which should lead to a nine-fold-amplified resulting wave that can also be generated in wave tanks.[1] Baker [6] has derived the exact KP–solution of the nine-fold amplified soliton resulting from the interaction of three solitons. The scalings introduced in this chapter may be applied to the analytical solution to simulate the nine-fold amplification under the Benney-Luke limit.

Some limits to the current model must also be highlighted. As already concluded in previous studies, the wave needs to propagate over a long distance (relative to its wavelength) in order to reach its maximal amplitude. Consequently, the numerical domain needs to be large and the mesh fine enough to estimate the wave crests accurately. This numerical requirement increases the computational time. A compromise between the accuracy of the simulations and the running time is therefore needed. This constraint is important because near the transition from Mach to regular reflection a slight change in the incident wave amplitude modifies dramatically the interaction parameter and consequently the predictions of the stem and reflected waves. Therefore, a careful analysis of the numerical results must be made. For the same reason, simulations for $\kappa \approx 1$ and large amplifications $\alpha_{\mathrm{w}} \approx 4$ are extremely difficult to obtain, since a slight change in the initial settings (a_{i}, ϵ and φ_{i}) modifies completely the behaviour of the resulting waves. Indeed, Li *et al.* [89] conjectured that the transition between Mach and regular reflection in the neighbourhood of $\kappa = 1$ might appear gradually and not as abruptly as expected from Miles' predictions (2.66). Moreover, the Benney-Luke model is valid under the assumption of weakly dispersive and weakly nonlinear waves. The threat of wave impact on ships being increased by the steepness of the waves, the modelling of Rogue–type waves for wave–structure–interaction applications would be improved by incorporating nonlinearity and dispersivity to the equations. In the next chapter, a new methodology is derived to simulate nonlinear, dispersive waves, including Rogue-type waves, in a tank where wave-structure interactions may be tested.

[1] O. Bokhove suggested this calculation to Y. Kodama, personal communication, who performed the calculation using the KP equation at the "Rogue waves" international workshop held at the Max Planck Institute in 2011, Dresden, Germany.

Chapter 3

Rogue-type waves in a deep-water tank

3.1 Introduction

An essential step in the simulation of freak-wave impact on marine structures is to maximize the amplitude and the steepness of the waves by combining the nonlinear and dispersion effects. While the long-wave dispersive Benney-Luke model derived in Chapter 2 was based on weak-nonlinearity and weak-dispersion parameters, we now consider a deep-water "numerical wave tank" in which nonlinear dispersive waves are driven by a vertical piston wavemaker. As in the in-house experimental basins of the Maritime Research Institute of Netherlands (MARIN), the wavemaker motion aims to emulate real sea states, including sophisticated wave-wave interactions. In order to assist in the design of experimental configurations at MARIN, the present numerical model was built to address several computational and industrial challenges.

First, the modelling of nonlinear water waves includes the capturing of the geometry of the free surface, at the air-water interface, which in mathematical terms is an *a priori* unknown boundary of a solution domain, for which most of the models in the literature are based on an adaptive mesh following the free-surface motion (see for instance [51, 93, 94, 95, 144]). An iterative update of the mesh is not practicable whithin the remit of our present aims since a major industrial requirement is the minimisation of the computational time. Instead, in section 3.2, we extend the method proposed by Engsig-Karup *et al.* [45], which consists of transforming the time-dependent free surface into a fixed boundary. We apply the same technique to the wavemaker boundary to solve the equations in a fully static numerical domain.

In contrast to the work of Engsig-Karup *et al.* [45], we use a variational approach in section 3.2 to derive weak formulations from Luke's variational principle [92]. Although Kim and Bai [81] obtained accurate simulations of two-dimensional potential-flow waves using this transformed variational approach, their model is not suitable for the above-mentioned applications, as 3D effects of the waves must be considered in order to simulate wave propagation in several directions, resulting in wave-wave or wave-structure interactions. Moreover, in [81], the waves travel in an unbounded domain, while the presence of wavemakers in our model will help industry to investigate wavemaker motions that, for the purposes of design and testing, generate specific waves in a target area.

By means of the variational approach, we obtain in section 3.3 a non-autonomous (wavemaker driven) space-discrete Hamiltonian system on which robust temporal integrators may be applied [53], complying with three essential computational requirements: stability and conservation of both mass and overall energy. Hence, in section 3.4, the 1^{st}–order symplectic Euler scheme is introduced along with the second-order Störmer-Verlet scheme; their numerical implementation with Firedrake [120, 9, 7, 71, 96] is explained in section 3.5. For both approaches, computational time is optimised through preconditioning of the Firedrake solvers in section 3.6: their performance in terms of computational speed and accuracy are compared in section 3.7, to assist the user in the choice of an optimised temporal-discretisation scheme appropriate to their bespoke application. In addition, a test of convergence is performed to verify the accuracy of the spatial discretisation method. In section 3.8, a comparison between the numerical free surface and the one measured at MARIN [24] shows that both temporal-integration schemes perform well in the simulation of extreme Rogue-type wave. A Fourier analysis of time series for both measurements and simulations confirms this observation. The final section sums up the strategies and achievements of the presented numerical model, augmented by a detailed presentation of current and future extensions for industrial applications. These improvements include the coupling of the simulated fluid motion to an absorbing beach, *i.e.* the objective of Chapter 4.

3.2 Variational nonlinear potential-flow model

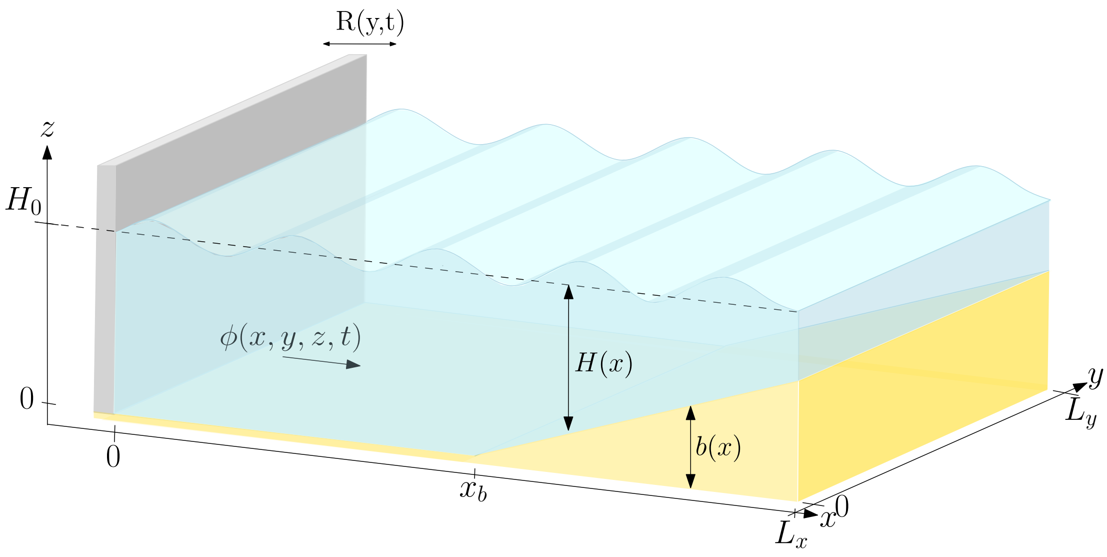


Figure 3.1: Schematic numerical wave tank. Waves are generated by a vertical piston wavemaker oscillating horizontally at $x = R(y,t)$ around $x = 0$. The depth at rest $H(x)$ varies in space due to the seabed topography $b(x)$ starting at $x = x_b$.

Water waves are often described by the Laplace equation for the velocity potential $\phi(x,y,z,t)$, augmented by two nonlinear boundary conditions (BCs): a kinematic BC which expresses that the boundary moves with the fluid and a dynamic BC, derived from the unsteady Bernoulli equation, which expresses the conservation of energy. These equations describe the dynamics of the total water depth $h(x,y,t) = H(x) + \eta(x,y,t)$, where $H(x)$ is the depth at rest and $\eta(x,y,t)$ is the surface deviation from $H(x)$, and of the velocity potential $\phi(x,y,z,t)$ which is defined such that the velocity field $\boldsymbol{u} = (u_x, u_y, u_z)$ may be expressed as $\boldsymbol{u} = \nabla\phi$. In this study, the nonlinear potential-flow equations

$$\nabla^2\phi = 0, \qquad \text{in } \Omega, \tag{3.3a}$$

$$\partial_t h + \nabla h \cdot \nabla\phi - \partial_z\phi = 0, \qquad \text{at } z = h, \tag{3.3b}$$

$$\partial_t\phi + \frac{1}{2}|\nabla\phi|^2 + g(h - H) = 0, \qquad \text{at } z = h, \tag{3.3c}$$

$$\partial_x\phi - \partial_y\phi\partial_y R = \partial_t R \qquad \text{at } x = R, \tag{3.3d}$$

where g is the gravitational constant, are obtained from Luke's variational principle [92] for an inviscid fluid:

$$0 = \delta \int_0^T \int_{\Omega_{x,y}} \int_0^{h(x,y,t)} \left[\partial_t \phi + \frac{1}{2}(\nabla\phi)^2 + g(z - H(x)) \right] \mathrm{d}z\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}t. \tag{3.4}$$

The horizontal domain $\Omega_{x,y} = \{R(y,t) \leq x \leq L_x; 0 \leq y \leq L_y\}$ is time-dependent due to the wavemaker boundary that moves around the position $x = 0$ as $R(y,t)$ (*cf.* Fig. 3.1). Similarly, the upper boundary of the domain, at $z = h(x,y,t)$, moves with the free surface $\eta(x,y,t)$. The numerical domain must therefore be discretised with a time-dependent mesh, with moving boundaries at $x = R(y,t)$ and $z = h(x,y,t)$. This constraint is not only costly in terms of computational time due to the update of the mesh at each time step, but it also requires one dealing with an unknown boundary, since the upper free-surface boundary $z = h(x,y,t)$ is part of the solution. Instead, a coordinate transform similar to the one introduced by Engsig-Karup *et al.* [45] is used to solve the equations on a constant domain, whose upper boundary is fixed so that no vertical mesh movement is required. Similarly, we introduce an additional coordinate transform in the x–direction to prevent the left boundary from moving with the wavemaker $R(y,t)$ in the numerical domain.

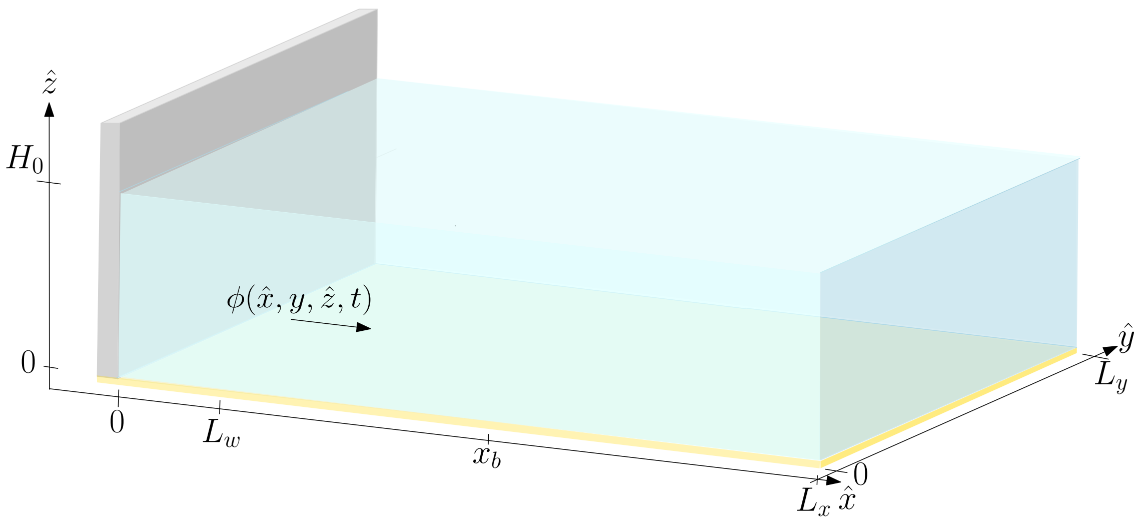


Figure 3.2: The fixed, computational domain as defined by $\hat{\Omega}$.

The resulting computational domain, as represented in Fig. 3.2, is defined as

$$\hat{\Omega} = \{0 \leq \hat{x} \leq L_x; 0 \leq \hat{y} \leq L_y; 0 \leq \hat{z} \leq H_0\}, \tag{3.5}$$

where $H_0 = \max_{x \in \Omega_{x,y}} H(x)$, and is obtained from the initial domain

$$\Omega = \{R(y,t) \leq x \leq L_x; 0 \leq y \leq L_y; 0 \leq z \leq h(x,y,t)\}$$

through the transformations

$$x \quad \rightarrow \quad \hat{x} = \frac{x - R(y,t)}{L_w - R(y,t)} L_w \Theta(L_w - x) + x\Theta(x - L_w), \tag{3.6a}$$

$$y \quad \rightarrow \quad \hat{y} = y, \tag{3.6b}$$

$$z \quad \rightarrow \quad \hat{z} = z\frac{H_0}{h(x,y,t)}, \tag{3.6c}$$

$$t \quad \rightarrow \quad \hat{t} = t. \tag{3.6d}$$

In Eq. (3.6a), Θ denotes the Heaviside function such that the coordinate transform is effective only in the area $x \in [R(y,t), L_w]$, where L_w is chosen to be sufficiently close to the wavemaker (about one wavelength): in this way, one can couple the water domain with a structure without the need to transform the structure in the x–direction away from the wavemaker. For example, in our case, the beginning of the seabed topography is set at $x = x_B > L_w$ so that $H(\hat{x}) = H(x)$ since the domain is transformed only where the topography is constant, with $H(x) = H_0$. In the case where $x_B \leq L_w$, then $H(x) \rightarrow H(\hat{x}(t))$ and additional terms arising from the time derivative of H must be considered, which complicates matters and we therefore exclude this situation. To avoid the division of the domain into $R(y,t) \leq x \leq L_w$ and $L_w \leq x \leq L$, we introduce the following variable:

$$\tilde{R}(x,y,t) = R(y,t)\Theta(L_w - x), \tag{3.7}$$

so that Eq. (3.6a) may be written as

$$x \rightarrow \hat{x} = \frac{x - \tilde{R}}{L_w - \tilde{R}} L_w. \tag{3.8}$$

Note that, due to the jump in the Heaviside function, the x–derivative of $\tilde{R}$ involves the Dirac function δ:

$$\partial_x \tilde{R} = R(y,t)\delta(L_w), \tag{3.9}$$

which vanishes once integrated over the x–domain:

$$\int_y \int_x \partial_x \tilde{R} \, \mathrm{d}x \, \mathrm{d}y = \int_y R(y,t) \int_x \delta(L_w) \, \mathrm{d}x \, \mathrm{d}y = \int_y \int_x R(y,t) \, \mathrm{d}x \, \mathrm{d}y, \tag{3.10}$$

since the integral of the Dirac function is equal to one. As only the weak form of the equations of motion will be implemented, the Heaviside function involved in (3.8) will not lead to any numerical issue. The y– and t– derivatives are also well defined, with

$$\partial_y \tilde{R} = \partial_y R \, \Theta(L_w - x) \qquad \text{and} \qquad \partial_t \tilde{R} = \partial_t R \, \Theta(L_w - x). \tag{3.11}$$

Equation (3.6c) transforms the vertical length to $[0, H_0]$ in the whole domain, thus allowing us to expand the solutions on a constant, prescribed mesh.

The spatial and temporal derivatives of the solutions $h(x,y,t)$ and $\phi(x,y,z,t)$ are then transformed as follows:

$$\partial_x \phi = \frac{L_w}{L_w - \tilde{R}} \left(\partial_{\hat{x}} \hat{\phi} - \frac{\hat{z}}{\hat{h}} \partial_{\hat{x}} \hat{h} \partial_{\hat{z}} \hat{\phi} \right), \tag{3.12a}$$

$$\partial_y \phi = \partial_{\hat{y}} \hat{\phi} - \frac{\hat{z}}{\hat{h}} \partial_{\hat{y}} \hat{h} \partial_{\hat{z}} \hat{\phi} + \frac{\hat{x} - L_w}{L_w - \tilde{R}} \partial_{\hat{y}} \tilde{R} \left(\partial_{\hat{x}} \hat{\phi} - \frac{\hat{z}}{\hat{h}} \partial_{\hat{x}} \hat{h} \partial_{\hat{z}} \hat{\phi} \right), \tag{3.12b}$$

$$\partial_z \phi = \frac{H_0}{\hat{h}} \partial_{\hat{z}} \hat{\phi}, \tag{3.12c}$$

$$\partial_t \phi = \partial_t \hat{\phi} - \frac{\hat{z}}{\hat{h}} \partial_t \hat{h} \partial_{\hat{z}} \hat{\phi} + \frac{\hat{x} - L_w}{L_w - \tilde{R}} \partial_t \tilde{R} \left(\partial_{\hat{x}} \hat{\phi} - \frac{\hat{z}}{\hat{h}} \partial_{\hat{x}} \hat{h} \partial_{\hat{z}} \hat{\phi} \right), \tag{3.12d}$$

$$\partial_x h = \frac{L_w}{L_w - \tilde{R}} \partial_{\hat{x}} \hat{h}, \tag{3.12e}$$

$$\partial_y h = \partial_{\hat{y}} \hat{h} + \frac{\hat{x} - L_w}{L_w - \tilde{R}} \partial_{\hat{y}} \tilde{R} \partial_{\hat{x}} \hat{h}, \tag{3.12f}$$

$$\partial_t h = \partial_t \hat{h} + \frac{\hat{x} - L_w}{L_w - \tilde{R}} \partial_t \tilde{R} \partial_{\hat{x}} \hat{h}, \tag{3.12g}$$

where $\hat{\phi} = \phi(\hat{x}, \hat{y}, \hat{z}; t)$ and $\hat{h} = h(\hat{x}, \hat{y}; t)$. For clarity, all hats are subsequently omitted. Substitution of the transformed coordinates into the variational principle (3.4) yields, after omitting the hats,

$$
\begin{aligned}
\delta\int_0^T\int_{\hat\Omega}\Big[&W\,(h\partial_t\phi - z\partial_t h\partial_z\phi) + (x-L_w)\partial_t\tilde R\,(h\partial_x\phi - z\partial_x h\partial_z\phi)\\
&+\frac{V}{2W}\left(h(\partial_x\phi)^2+\frac{z^2}{h}(\partial_x h)^2(\partial_z\phi)^2 - 2z\partial_x h\partial_x\phi\partial_z\phi\right)\\
&+\frac{W}{2}\left(h(\partial_y\phi)^2+\frac{z^2}{h}(\partial_y h)^2(\partial_z\phi)^2 - 2z\partial_y h\partial_y\phi\partial_z\phi\right)\\
&+U\Big(h\partial_x\phi\partial_y\phi - z\partial_x h\partial_y\phi\partial_z\phi - z\partial_y h\partial_x\phi\partial_z\phi + \frac{z^2}{h}\partial_x h\partial_y h(\partial_z\phi)^2\Big)\\
&+\frac{WH_0^2}{2h}(\partial_z\phi)^2 + gWh(\frac{h}{H_0}z - H)\Big]\frac{1}{H_0L_w}\mathrm{d}z\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t = 0,
\end{aligned}
\tag{3.13}
$$

where

$$U(x,y,t) = (x-L_w)\,\partial_y\tilde R, \tag{3.14a}$$

$$V(x,y,t) = \left(L_w^2 + (x-L_w)^2(\partial_y\tilde R)^2\right), \tag{3.14b}$$

$$W(x,y,t) = \left(L_w - \tilde R\right). \tag{3.14c}$$

After multiplication by H_0L_w and integration by parts in time of the first term, in z of the second and fourth terms, and in x of the third term we obtain:

$$
\begin{aligned}
\delta\int_0^T\Bigg\{\int_{\hat\Omega_{x,y}}\Bigg[\int_0^{H_0}\Bigg[&\frac{V}{2W}\left(h(\partial_x\phi)^2+\frac{z^2}{h}(\partial_x h)^2(\partial_z\phi)^2 - 2z\partial_x h\partial_x\phi\partial_z\phi\right)\\
&+\frac{W}{2}\left(h(\partial_y\phi)^2+\frac{z^2}{h}(\partial_y h)^2(\partial_z\phi)^2 - 2z\partial_y h\partial_y\phi\partial_z\phi\right)\\
&+U\Big(h\partial_x\phi\partial_y\phi - z\partial_x h\partial_y\phi\partial_z\phi - z\partial_y h\partial_x\phi\partial_z\phi\\
&\quad+\frac{z^2}{h}\partial_x h\partial_y h(\partial_z\phi)^2\Big) + \frac{WH_0^2}{2h}(\partial_z\phi)^2\Bigg]\mathrm{d}z\\
+H_0\Big(gWh(\frac{1}{2}h - H) - \phi&\left(W\partial_t h + (x-L_w)\,\partial_t\tilde R\,\partial_x h\right)\Big)_{z=H_0}\Bigg]\mathrm{d}x\,\mathrm{d}y\\
+\int_0^{L_y}\int_0^{H_0}\left(L_w\,\partial_t\tilde R\,\phi\,h\right)_{x=0}&\mathrm{d}z\mathrm{d}y\Bigg\}\mathrm{d}t = 0,
\end{aligned}
\tag{3.15}
$$

where $\hat\Omega_{x,y}$ denotes the fixed horizontal domain, that is $\hat\Omega_{x,y} = \{0\le x\le L_x; 0\le y\le L_y\}$. The transformed Laplace equation, the kinematic and dynamic BCs, the wavemaker and Neumann BCs may be derived from Eq. (3.15) through variations of both h and ϕ, with temporal end-point conditions $\delta h(x,y,0)=0$ and $\delta h(x,y,T)=0$. The variations yield

$$
\begin{aligned}
\delta\phi :\ \Bigg[&(x-L_w)\left(\frac{2}{W}(\partial_y\tilde{R})^2+\partial_{yy}\tilde{R}\right)(z\partial_x h\partial_z\phi-h\partial_x\phi)-W\frac{H_0^2}{h}\partial_{zz}\phi\\
&-\frac{V}{W}\Big(h\partial_{xx}\phi+\frac{z}{h}(\partial_x h)^2\left(2\partial_z\phi+z\partial_{zz}\phi\right)-z\left(\partial_z\phi\partial_{xx}h+2\partial_x h\partial_{xz}\phi\right)\Big)\\
&-W\bigg(h\partial_{yy}\phi+\frac{z}{h}(\partial_y h)^2\left(2\partial_z\phi+z\partial_{zz}\phi\right)-z\left(\partial_z\phi\partial_{yy}h+2\partial_y h\partial_{yz}\phi\right)\bigg)\\
&-2U\Big(h\partial_{xy}\phi+\frac{z}{h}\partial_x h\partial_y h\left(2\partial_z\phi+z\partial_{zz}\phi\right)\\
&\qquad-z\left(\partial_z\phi\partial_{xy}h+\partial_x h\partial_{yz}\phi+\partial_y h\partial_{xz}\phi\right)\Big)\Bigg]=0 \qquad\qquad \text{in } \Omega,
\end{aligned} \tag{3.16a}
$$

$$
\begin{aligned}
\delta\phi_{z=H_0} :\ \Bigg[&\frac{V}{W}\left(\frac{H_0}{h}(\partial_x h)^2\partial_z\phi-\partial_x h\partial_x\phi\right)-(x-L_w)\partial_t\tilde{R}\partial_x h\\
&+W\left(\frac{H_0}{h}(\partial_z\phi)\left(1+H_0(\partial_y h)^2\right)-(\partial_y h\partial_y\phi+\partial_t h)\right)\\
&+U\left(-\partial_x h\partial_y\phi-\partial_y h\partial_x\phi+2\frac{H_0}{h}\partial_x h\partial_y h\partial_z\phi\right)\Bigg]=0 \qquad \text{at } z=H_0,
\end{aligned} \tag{3.16b}
$$

$$
\begin{aligned}
\delta h_{H_0} :\ \Bigg[&\left(g(h-H)+\partial_t\phi+\frac{1}{2}(\partial_y\phi)^2-\frac{1}{2}\frac{H_0^2}{h^2}(\partial_z\phi)^2\left(1+(\partial_y h)^2\right)\right)\\
&+\frac{(x-L_w)}{W}\left[\partial_t\tilde{R}\partial_x\phi+\partial_y\tilde{R}\left(\partial_x\phi\partial_y\phi-\frac{H_0^2}{h^2}\partial_x h\partial_y h(\partial_z\phi)^2\right)\right]\\
&+\frac{V}{2W^2}\left((\partial_x\phi)^2-\frac{H_0^2}{h^2}(\partial_x h)^2(\partial_z\phi)^2\right)\Bigg]=0, \qquad \text{at } z=H_0,
\end{aligned} \tag{3.16c}
$$

$$
\begin{aligned}
\delta\phi_{x=0} :\ &\frac{L_w+L_w(\partial_y\tilde{R})^2}{W}\left(\frac{z}{h}\partial_x h\partial_z\phi-\partial_x\phi\right)\\
&+\partial_y\tilde{R}\left(\partial_y\phi-\frac{z}{h}\partial_y h\partial_z\phi\right)=-\partial_t\tilde{R} \qquad \text{at } x=0,
\end{aligned} \tag{3.16d}
$$

$$
\delta\phi_{x=L_x} :\ L_w\left(h\partial_x\phi-z\partial_x h\partial_z\phi\right)=0 \qquad \text{at } x=L_x, \tag{3.16e}
$$

$$
\delta\phi_{z=H_0} :\ \frac{WH_0^2}{h}\partial_z\phi=0 \qquad \text{at } z=0, \tag{3.16f}
$$

$$
\delta\phi_{y=0,L_y} :\ W\left(z\partial_y h\partial_z\phi-h\partial_y\phi\right)+U\left(z\partial_x h\partial_z\phi-h\partial_x\phi\right)=0 \qquad \text{at } y=0,L_y. \tag{3.16g}
$$

As a check, shown in Appendix B.1, the same equations are obtained by transforming the initial Laplace equation and the kinematic and dynamic boundary conditions (3.3a-d). The diagram 3.3 summarises the two ways of obtaining Eqs. (3.16).

$$\begin{tikzcd}
\boxed{\text{Variational Principle (3.4)}} \arrow[r, "\text{Scaling (3.6)}"] \arrow[d, "\text{Variations}"'] & \boxed{\text{Transformed variational principle (3.15)}} \arrow[d, "\text{Variations}"] \\
\boxed{\text{Equations 3.3}} \arrow[r, "\text{Scaling (3.6)}"] & \boxed{\text{Transformed equations (3.16) and (B.9)}}
\end{tikzcd}$$

Figure 3.3: Diagram showing two ways (blue and pink paths) to obtain the transformed Euler equations.

3.3 Spatial discretisation strategies

3.3.1 Updating the vertical structure

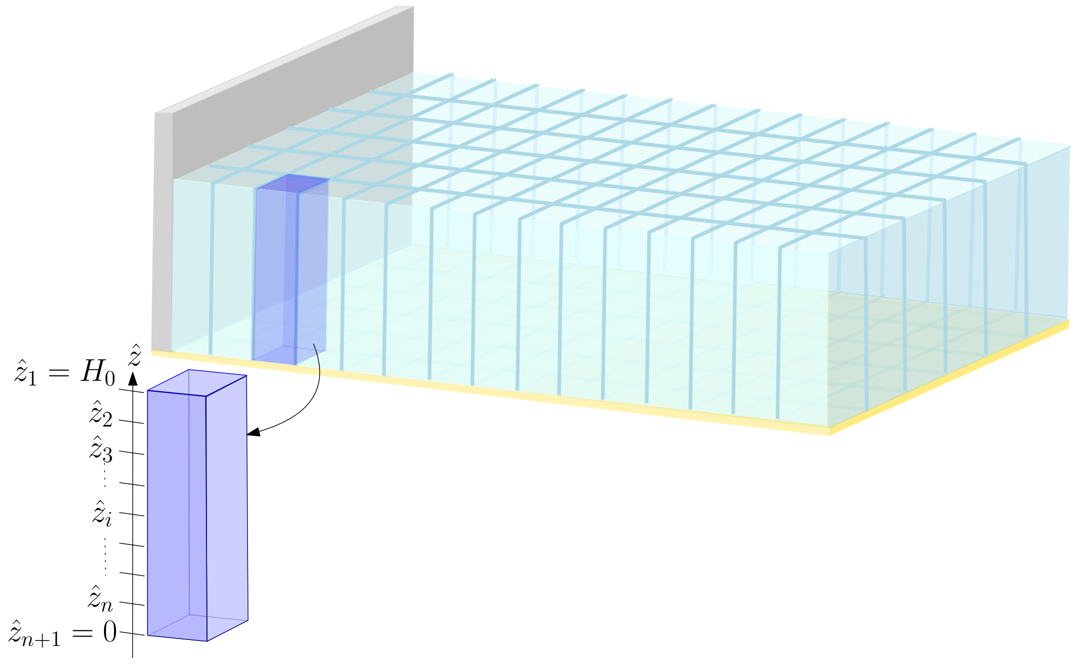


Figure 3.4: Discretised 3D fixed domain $\hat{\Omega}_d$. The mesh contains $N_x \times N_y$ elements in the horizontal plane, and one vertical element on which the velocity potential is expanded with high order expansions in order to eliminate the z–dependency of the weak formulations.

To solve Eqs. (3.16), the domain and the equations must be discretised in space. The package Firedrake [120, 9, 7, 71, 96] is used to solve the weak formulations with the finite-element method (see, for example, [121]). This automated system discretises the equations in space internally, based on user-defined settings regarding the mesh and the expansions to use. In Eq. (3.16), the functions h and ϕ are updated at the surface only, through Eq. (3.16b) and Eq. (3.16c) respectively. While h is only defined at the surface, the velocity potential ϕ also evolves in depth, and its subsurface values are slaves of its surface evolution through the (transformed) Laplace equation Eq. (3.16a). In order to update ϕ both at the surface and in the interior, we therefore need to distinguish its surface and interior evaluations. For this purpose, we use the Schur-complement method which is based on the Dirichlet-to-Neumann (DtN) operator [34] and aims to decompose the vertical domain into non-overlapping subdomains so that the subsurface subdomains may later be eliminated. Essentially, we discretise the 3D transformed domain $\hat{\Omega}$ with $N_x \times N_y$ quadrilateral elements in the horizontal plane, but only one vertical element, on which the velocity potential is expanded with high-order (n_z) expansions as

$$\phi(x, y, z, t) = \psi_i(x, y, t)\tilde{\varphi}_i(z), \tag{3.18}$$

where the Einstein summation convention is used for repeated indices $i \in [1, n_z + 1]$.

Figure (3.4) illustrates the discrete mesh. Substitution of Eq. (3.18) into the variational principle (3.15) enables us to separate z–integrals from the x– and y–integrals as follows:

$$\begin{aligned}
\delta \int_0^T \Bigg\{ \int_{\hat{\Omega}_{x,y}} \Bigg[& \frac{V}{2W}\left(h\partial_x\psi_i\partial_x\psi_j\tilde{M}_{ij} + \frac{1}{h}(\partial_x h)^2\psi_i\psi_j\tilde{S}_{ij} - 2\psi_i\partial_x h\partial_x\psi_j\tilde{D}_{ji}\right) \\
& + \frac{W}{2}\left(h\partial_y\psi_i\partial_y\psi_j\tilde{M}_{ij} + \frac{1}{h}(\partial_y h)^2\psi_i\psi_j\tilde{S}_{ij} - 2\psi_i\partial_y h\partial_y\psi_j\tilde{D}_{ji}\right) \\
& + U\Bigg(h\partial_x\psi_i\partial_y\psi_j\tilde{M}_{ij} + \frac{1}{h}\partial_x h\partial_y h\psi_i\psi_j\tilde{S}_{ij} \\
& \qquad - \psi_i\partial_x h\partial_y\psi_j\tilde{D}_{ji} - \psi_i\partial_y h\partial_x\psi_j\tilde{D}_{ji}\Bigg) \\
& + \frac{WH_0^2}{2h}\psi_i\psi_j\tilde{A}_{ij} + H_0\Big(gWh(\frac{1}{2}h - H) \\
& \qquad - \psi_i\tilde{\varphi}_i(H_0)\left((x - L_w)\partial_t R\partial_x h + W\partial_t h\right)\Big)_{z=H_0}\Bigg] \mathrm{d}x\,\mathrm{d}y \\
+ \int_0^{L_y} & \left(L_w\,\partial_t\tilde{R}\,h\,\psi_i\,\tilde{I}_i\right)_{x=0}\mathrm{d}y \Bigg\}\,\mathrm{d}t = 0,
\end{aligned} \tag{3.19}$$

in which the matrices $\tilde{M}$, $\tilde{D}$, $\tilde{A}$, $\tilde{S}$ and $\tilde{I}$ are defined as

$$\tilde{M}_{ij} = \int_0^{H_0} [\tilde{\varphi}_i\tilde{\varphi}_j]\,\mathrm{d}z, \quad (3.20a) \qquad \tilde{D}_{ij} = \int_0^{H_0} [z\tilde{\varphi}_i d_z\tilde{\varphi}_j]\,\mathrm{d}z, \quad (3.20d)$$

$$\tilde{A}_{ij} = \int_0^{H_0} [d_z\tilde{\varphi}_i d_z\tilde{\varphi}_j]\,\mathrm{d}z, \quad (3.20b) \qquad \tilde{S}_{ij} = \int_0^{H_0} \left[z^2 d_z\tilde{\varphi}_i d_z\tilde{\varphi}_j\right]\mathrm{d}z. \quad (3.20e)$$

$$\tilde{I}_i = \int_0^{H_0} [\tilde{\varphi}_i]\,\mathrm{d}z, \quad (3.20c)$$

Equation (3.19) holds the dynamics of the evolution of h and ϕ in the horizontal plane, while the vertical components of ϕ are considered as coefficients through the constant matrices (3.20). These matrices are evaluated numerically with the python package Sympy [101]: more details are given in the tutorials in Chapter 6. The next section explains what strategic indexing is used to distinguish the surface and interior nodal evaluations of ϕ.

3.3.2 Expansion of subsurface velocity potential

The vertical component of the velocity potential is expanded with a Lagrange polynomial of order n_z as

$$\tilde{\varphi}_i(z) = \prod_{\substack{k=1\\k\neq i}}^{n_z+1} \frac{z - z_k}{z_i - z_k}, \tag{3.21}$$

with z_i the discrete vertical coordinate, defined for all $i \in [1, n_z + 1]$ as

$$z_i = \frac{H_0}{n_z}(n_z + 1 - i). \tag{3.22}$$

The value of n_z must be set by the user depending on the required vertical resolution. Similarly, the linear distribution Eq. (3.22) may be changed to non-uniform points, such as exponential distribution. From Eq. (3.21), the polynomial $\tilde{\varphi}_i$, with $i \in [1, n_z + 1]$, is defined so that

$$\tilde{\varphi}_i(z_k) = \delta_{ik} = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{if } k \neq i. \end{cases} \tag{3.23}$$

Our strategy is to set the index $i = 1$ at the surface, and use $i' \in [2, n_z + 1]$ in the interior layers; that is,

$$\begin{cases} z_1 = H_0, \\ 0 \leq z_{i'} < H_0, \end{cases} \tag{3.24}$$

so that, $\forall i' \in [2, n_z+1]$,

$$\begin{cases} \tilde{\varphi}_{i'}(H_0) = 0, \\ \tilde{\varphi}_{i'}(z_{i'}) = 1. \end{cases} \quad \text{and} \quad \begin{cases} \tilde{\varphi}_1(H_0) = 1, \\ \tilde{\varphi}_1(z_{i'}) = 0, \end{cases} \tag{3.25}$$

As a consequence, the surface and interior evaluations of the velocity potential may be distinguished through

$$\phi(x,y,z,t) = \begin{cases} \psi_1(x,y,t) \quad \text{at the surface } z = z_1 = H_0, \\ \psi_{i'}(x,y,t) \quad \text{on the horizontal plane } z = z_{i'} < H_0, \\ \displaystyle\sum_{j=1}^{n_z+1} \psi_j(x,y,t)\tilde{\varphi}_j(z) \text{ for } z \neq z_i, \forall i \in [1, n_z+1]. \end{cases} \tag{3.26}$$

Substitution of Eq. (3.26) into the variational principle (3.19) leads to

$$\begin{aligned}
\delta \int_0^T \Bigg\{ \int_{\hat{\Omega}_{x,y}} \Bigg[& \frac{V}{2W} h \Big[(\partial_x\psi_1)^2 \tilde{M}_{11} + \partial_x\psi_{i'} \Big(2\tilde{M}_{i'1}\partial_x\psi_1 + \tilde{M}_{i'j'}\partial_x\psi_{j'} \Big) \Big] \\
& + \frac{Wh}{2} \Big[(\partial_y\psi_1)^2 \tilde{M}_{11} + \partial_y\psi_{i'} \Big(2\tilde{M}_{i'1}\partial_y\psi_1 + \tilde{M}_{i'j'}\partial_y\psi_{j'} \Big) \Big] \\
& + Uh \Big[\partial_x\psi_1 \Big(\partial_y\psi_1\tilde{M}_{11} + \partial_y\psi_{j'}\tilde{M}_{1j'} \Big) + \partial_x\psi_{i'} \Big(\partial_y\psi_1\tilde{M}_{i'1} + \partial_y\psi_{j'}\tilde{M}_{i'j'} \Big) \Big] \\
& + \frac{1}{h} \Big[\frac{V}{2W}(\partial_x h)^2 + \frac{W}{2}(\partial_y h)^2 + U\partial_x h \partial_y h \Big] \\
& \quad \times \Big[\psi_1^2 \tilde{S}_{11} + \psi_{i'} \Big(2\tilde{S}_{i'1}\psi_1 + \tilde{S}_{i'j'}\psi_{j'} \Big) \Big] \\
& - \Big[\frac{V}{W}\partial_x h + U\partial_y h \Big] \\
& \quad \times \Big[\partial_x\psi_1 \Big(\tilde{D}_{11}\psi_1 + \tilde{D}_{1i'}\psi_{i'} \Big) + \partial_x\psi_{i'} \Big(\tilde{D}_{i'1}\psi_1 + \tilde{D}_{i'j'}\psi_{j'} \Big) \Big] \\
& - [W\partial_y h + U\partial_x h] \\
& \quad \times \Big[\partial_y\psi_1 \Big(\tilde{D}_{11}\psi_1 + \tilde{D}_{1i'}\psi_{i'} \Big) + \partial_y\psi_{i'} \Big(\tilde{D}_{i'1}\psi_1 + \tilde{D}_{i'j'}\psi_{j'} \Big) \Big] \\
& + \frac{WH_0^2}{2h} \Big[\psi_1^2 \tilde{A}_{11} + \psi_{i'} \Big(2\tilde{A}_{i'1}\psi_1 + \tilde{A}_{i'j'}\psi_{j'} \Big) \Big] \\
& + H_0 \Big[gWh(\frac{1}{2}h - H) - \psi_1 \Big((x - L_w)\partial_t\tilde{R}\partial_x h + W\partial_t h \Big) \Big] \Bigg] \mathrm{d}x\,\mathrm{d}y \\
+ \int_0^{L_y} \Big(L_w\, \partial_t\tilde{R}\, h \Big(\psi_1\tilde{I}_1 + \psi_{i'}\tilde{I}_{i'} \Big) & \Big)_{x=0} \mathrm{d}y \Bigg\} \mathrm{d}t = 0.
\end{aligned} \tag{3.27}$$

The technique described above to distinguish the surface and interior evaluations leads to the variational principle (3.27) for $n_z + 2$ unknowns: the depth h, the velocity potential at the surface ψ_1 and the velocity potential in the interior $\psi_{i'}$ for $i' \in [2, n_z + 1]$. The variations of (3.27) with respect to each $\psi_{i'}$ will lead to n_z extra equations describing the evolution of the velocity potential on each interior layer of the domain. That way, we are able to update $\phi(x, y, z, t)$ in the three-dimensional domain. In the next section, the finite-element method is used to expand the variables h, ψ_1 and $\psi_{i'}$ in the horizontal plane.

3.3.3 Finite-element method in the horizontal plane

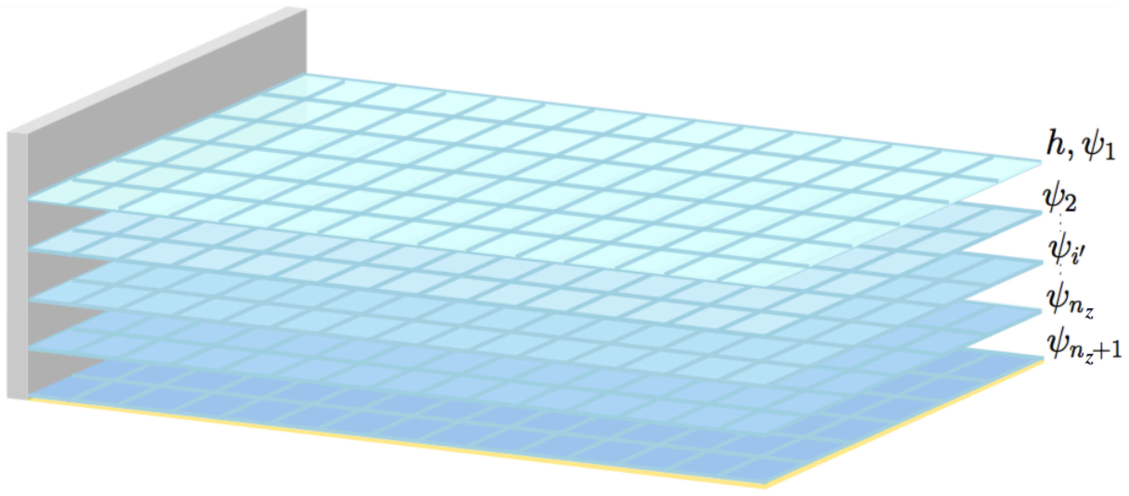


Figure 3.5: Discrete domain for solving the transformed Euler equations with the finite-element method. The unknowns are expanded in each horizontal plane with continuous Galerkin expansions.

A finite-element method based on continuous Galerkin expansions is used to discretise the variables on each horizontal layer. While $h(x, y, t)$ and $\psi_1(x, y, t)$ evolve on the surface plane, $\psi_{i'}(x, y, t)$ evolves on the interior layers (*cf.* Fig.3.5). Each layer is discretised with $N_h = N_x \times N_y$ elements, on which the solutions are evaluated through their temporal coefficients and basis functions as

$$\begin{aligned} h_h &= h_k(t)\varphi_k(x,y), \\ \psi_{1_h} &= \psi_{1k}(t)\varphi_k(x,y), \\ \psi_{i'h} &= \psi_{i'k}(t)\varphi_k(x,y), \end{aligned} \tag{3.28}$$

where the Einstein implicit summation convention for repeated index $k \in [1, N_h]$ is used. Substitution of the finite-element expansions (3.28) into the variational principle (3.27) leads to the space-discrete time-continuous variational principle. In matrix-tensor form it reads

$$\begin{aligned} \delta\int_0^T \Bigg\{ & \frac{h_k}{2}\Big[\Lambda_{kqm}\psi_{1q}\left(\tilde{M}_{11}\psi_{1m} + \tilde{M}_{i'1}\psi_{i'm}\right) + \Lambda_{kmq}\psi_{i'm}\left(\tilde{M}_{i'1}\psi_{1q} + \tilde{M}_{i'j'}\psi_{j'q}\right)\Big] \\ & -h_k\Big[\Gamma_{mkq}\psi_{1q}\left(\tilde{D}_{11}\psi_{1m} + \tilde{D}_{1i'}\psi_{i'm}\right) + \Gamma_{qkm}\psi_{i'm}\left(\tilde{D}_{i'1}\psi_{1q} + \tilde{D}_{i'j'}\psi_{j'q}\right)\Big] \\ & +\frac{h_l h_p}{2h_k}\Upsilon_{kmqlp}\Big[\psi_{1q}\tilde{S}_{11}\psi_{1m} + \psi_{i'm}\left(2\tilde{S}_{i'1}\psi_{1q} + \tilde{S}_{i'j'}\psi_{j'q}\right)\Big] \\ & +\frac{H_0^2}{2h_k}J_{kmq}\left[\psi_{1q}\tilde{A}_{11}\psi_{1m} + 2\psi_{i'm}\tilde{A}_{i'1}\psi_{1q} + \psi_{i'm}\tilde{A}_{i'j'}\psi_{j'q}\right] \\ & +L_w h_k\left[X_{kq}\psi_{1q}\tilde{I}_1 + X_{km}\psi_{i'm}\tilde{I}_{i'}\right] \\ & +H_0\left(gh_k(\frac{1}{2}h_l M_{kl} - HI_k) - \psi_{1q}N_{qk}h_k - M_{kq}\psi_{1q}\frac{dh_k}{dt}\right)_{z=H_0}\Bigg\}\,\mathrm{d}t = 0, \end{aligned} \tag{3.29}$$

where

$$N_{qk} = \int_{\hat{\Omega}_{x,y}} \left[(x - L_w)\partial_t\tilde{R}\varphi_q\partial_x\varphi_k\right]\mathrm{d}x\mathrm{d}y, \tag{3.30a}$$

$$M_{kl} = \int_{\hat{\Omega}_{x,y}} \left[W\varphi_k\varphi_l\right]\,\mathrm{d}x\,\mathrm{d}y, \tag{3.30b}$$

$$I_k = \int_{\hat{\Omega}_{x,y}} \left[W\varphi_k\right]\,\mathrm{d}x\,\mathrm{d}y, \tag{3.30c}$$

$$J_{kmq} = \int_{\hat{\Omega}_{x,y}} \left[\frac{W}{\varphi_k}\varphi_m\varphi_q\right]\mathrm{d}x\mathrm{d}y, \tag{3.30d}$$

$$X_{kq} = \int_0^{L_y} \left[\partial_t\tilde{R}\varphi_k\varphi_q\right]_{x=0}\,\mathrm{d}y, \tag{3.30e}$$

$$\Gamma_{kmq} = A_{kmq} + B_{kmq} + D_{kmq} + D_{kqm}, \tag{3.30f}$$

$$\Lambda_{kmq} = A_{kmq} + B_{kmq} + 2D_{kmq}, \tag{3.30g}$$

$$\Upsilon_{kmqlp} = E_{kmqlp} + F_{kmqlp} + 2G_{kmqlp}, \tag{3.30h}$$

with

$$
\begin{aligned}
A_{kmq} &= \int_{\hat{\Omega}_{x,y}} \left[\frac{V}{W}\varphi_k\partial_x\varphi_m\partial_x\varphi_q\right] \mathrm{d}x\,\mathrm{d}y, & E_{kmqlp} &= \int_{\hat{\Omega}_{x,y}} \left[\frac{V}{W}\frac{1}{\varphi_k}\varphi_m\varphi_q\partial_x\varphi_l\partial_x\varphi_p\right] \mathrm{d}x\,\mathrm{d}y,\\
B_{kmq} &= \int_{\hat{\Omega}_{x,y}} \left[W\varphi_k\partial_y\varphi_m\partial_y\varphi_q\right] \mathrm{d}x\,\mathrm{d}y, & F_{kmqlp} &= \int_{\hat{\Omega}_{x,y}} \left[\frac{W}{\varphi_k}\varphi_m\varphi_q\partial_y\varphi_l\partial_y\varphi_p\right] \mathrm{d}x\,\mathrm{d}y,\\
D_{kmq} &= \int_{\hat{\Omega}_{x,y}} \left[U\varphi_k\partial_x\varphi_m\partial_y\varphi_q\right] \mathrm{d}x\,\mathrm{d}y, & G_{kmqlp} &= \int_{\hat{\Omega}_{x,y}} \left[U\frac{1}{\varphi_k}\varphi_m\varphi_q\partial_x\varphi_l\partial_y\varphi_p\right] \mathrm{d}x\,\mathrm{d}y.
\end{aligned}
$$

In the next section, the space-discrete time-continuous variational principle (3.29) is used to derive robust time integrators. These time integrators are then applied directly to the space-continuous variational principle (3.27) and solved using Firedrake [120, 9, 7, 71, 96], as explained in section 3.5.

3.4 Temporal discretisation schemes

One advantage of using the variational approach is that the variational principle may be written in Hamiltonian form, for which robust and symplectic temporal integrators exist [53]. In this section, we write Eq. (3.29) in Hamiltonian form in order to apply two temporal integrators of first and second order, following the method described in [53].

3.4.1 Hamiltonian dynamics

The space-discrete time-continuous variational principle (3.29) may be written in Hamiltonian form in terms of the coordinate $h_k(t)$ and the momentum $p_{1k} = M_{kq}(t)\psi_{1q}(t)$ as follows:

$$0 = \delta \int_0^T p_{1k}\frac{dh_k}{dt} - H\left(\boldsymbol{h}, \boldsymbol{p_1}, \boldsymbol{\psi_{i'}}, t\right)\,\mathrm{d}t, \tag{3.31}$$

with H the Hamiltonian defined as

$$\begin{aligned}
H(\boldsymbol{h},\boldsymbol{p_1},\boldsymbol{\psi_{i'}},t) = \Bigg\{ & \frac{h_k}{2}\Big[\Lambda_{kqm}M^{-1}_{rq}p_{1r}\left(\tilde{M}_{11}M^{-1}_{sm}p_{1s}+\tilde{M}_{1i'}\psi_{i'm}\right) \\
& \qquad + \Lambda_{kmq}\psi_{i'm}\left(\tilde{M}_{i'1}M^{-1}_{rq}p_{1r}+\tilde{M}_{i'j'}\psi_{j'q}\right)\Big] \\
& -h_k\Big[M^{-1}_{rq}p_{1r}\Gamma_{mkq}\left(\tilde{D}_{11}M^{-1}_{sm}p_{1s}+\tilde{D}_{1i'}\psi_{i'm}\right) \\
& \qquad + \psi_{i'm}\Gamma_{qkm}\left(\tilde{D}_{i'1}M^{-1}_{rq}p_{1r}+\tilde{D}_{i'j'}\psi_{j'q}\right)\Big] \\
& +\frac{h_lh_p}{2h_k}\Upsilon_{kmqlp}\Big[\psi_{i'm}\tilde{S}_{i'j'}\psi_{j'q}+M^{-1}_{rq}p_{1r}\left(\tilde{S}_{11}M^{-1}_{sm}p_{1s}+2\psi_{i'm}\tilde{S}_{i'1}\right)\Big] \\
& +\frac{H_0^2}{2h_k}J_{kmq}\Big[M^{-1}_{rq}p_{1r}\left(\tilde{A}_{11}M^{-1}_{sm}p_{1s}+2\psi_{i'm}\tilde{A}_{i'1}\right)+\psi_{i'm}\tilde{A}_{i'j'}\psi_{j'q}\Big] \\
& +L_wh_k\Big[X_{kq}M^{-1}_{rq}p_{1r}\tilde{I}_1+X_{km}\psi_{i'm}\tilde{I}_{i'}\Big] \\
& +H_0\left(gh_k(\frac{1}{2}h_lM_{kl}-HI_k)-M^{-1}_{rq}p_{1r}N_{qk}h_k\right)\Bigg\}.
\end{aligned} \tag{3.32}$$

The variations of Eq. (3.31) with respect to $\boldsymbol{p_1}$, $\boldsymbol{h}$ and $\psi_{i'}$ lead to the following system of equations:

$$\delta p_k: \quad \frac{dh_k}{dt}=\frac{\partial H(\boldsymbol{h},\boldsymbol{p_1},\boldsymbol{\psi_i},t)}{\partial \boldsymbol{p_1}}, \tag{3.33a}$$

$$\delta h_k: \quad \frac{dp_k}{dt}=-\frac{\partial H(\boldsymbol{h},\boldsymbol{p_1},\boldsymbol{\psi_i},t)}{\partial \boldsymbol{h}}, \tag{3.33b}$$

$$\delta \psi_{i'm}: \quad \frac{\partial H(\boldsymbol{h},\boldsymbol{p_1},\boldsymbol{\psi_i},t)}{\partial \boldsymbol{\psi_{i'}}}=0. \tag{3.33c}$$

The spatial discretisation strategies described in Section 3.3 lead to the extra equation (3.33c) used to update the velocity potential $\psi_{i'}$ in the interior. Equation (3.33c) may be written in explicit form as

$$\begin{aligned}
\psi_{j'q} = -\Bigg[& h_k\left(\Gamma_{kqm}\tilde{M}_{i'j'}-\Gamma_{qkm}\tilde{D}_{i'j'}-\Gamma_{mkq}\tilde{D}_{j'i'}\right) \\
& +\frac{1}{h_k}\left(h_lh_p\Upsilon_{kmqlp}\tilde{S}_{i'j'}+H_0^2J_{kmq}\tilde{A}_{i'j'}\right)\Bigg]^{-1} \\
\times\Bigg\{ & \Bigg[h_k\left(\Gamma_{kqm}\tilde{M}_{i'1}-\Gamma_{qkm}\tilde{D}_{i'1}-\Gamma_{mkq}\tilde{D}_{1i'}\right) \\
& +\frac{1}{h_k}\left(h_lh_p\Upsilon_{kmqlp}\tilde{S}_{i'1}+H_0^2J_{kmq}\tilde{A}_{i'1}\right)\Bigg]M^{-1}_{rq}p_{1r}-h_kL_wX_{km}\tilde{I}_{i'}\Bigg\},
\end{aligned} \tag{3.34}$$

which is a linear expression of $\boldsymbol{\psi_{i'}}$ in terms of $\boldsymbol{h}$, $\boldsymbol{p_1}$ and W resulting from the discretization of the linear Laplace equation. One option is thus to eliminate the interior values of the velocity potential

$\psi_{\boldsymbol{i'}}$ by substituting Eq. (3.34) into the Hamiltonian (3.32), so that the initial variational principle Eq. (3.31) may be expressed in terms of an auxiliary Hamiltonian $\tilde{H}$ that depends on $\boldsymbol{h}$, $\boldsymbol{p_1}$ and W only:

$$0 = \delta \int_0^T p_{1k} \frac{dh_k}{dt} - \tilde{H}\left(\boldsymbol{h}, \boldsymbol{p_1}, W(t)\right) \, \mathrm{d}t. \tag{3.35}$$

Variations of Eq. (3.35) with respect to $\boldsymbol{h}$ and $\boldsymbol{p_1}$ would then reduce Eqs. (3.33) to a system of two equations:

$$\frac{dh_k}{dt} = \frac{\partial \tilde{H}}{\partial \boldsymbol{p_1}}, \qquad \text{and} \qquad \frac{dp_{1k}}{dt} = -\frac{\partial \tilde{H}}{\partial \boldsymbol{h}}. \tag{3.36}$$

However, the expression of $\psi_{\boldsymbol{i'}}$ in Eq. (3.34) involves many terms and the resulting Hamiltonian $\tilde{H}(\boldsymbol{h}, \boldsymbol{p_1}, W)$ would include many matrix products and matrix inverses. Instead, another option is to eliminate $\psi_{\boldsymbol{i'}}(\boldsymbol{h}, \boldsymbol{p_1}, W)$ by solving Eq. (3.33c), that is Eq. (3.34), simultaneously with Eqs. (3.33a-b). This option is equivalent to solving the system of equations (3.36), provided that $\boldsymbol{h}$, $\boldsymbol{p_1}$ and W are evaluated at the same temporal evaluations in Eqn (3.34) and in the Hamiltonian used in Eqs. (3.33a-b).

For N discontinuous time intervals $[t^n, t^{n+1}]$, Gagarina *et al.* [53] discretised a variational principle of the form

$$0 = \delta \int_0^T \boldsymbol{p} \frac{d\boldsymbol{q}}{dt} - H(\boldsymbol{p}, \boldsymbol{q}) \, \mathrm{d}t, \tag{3.37}$$

in terms of the polynomial approximations $\boldsymbol{p^\tau}$ and $\boldsymbol{q^\tau}$ of $\boldsymbol{p}$ and $\boldsymbol{q}$ respectively, as

$$0 = \delta \left\{ \sum_{n=0}^{N-1} \int_{t^n}^{t^{n+1}} \left(\boldsymbol{p^\tau} \frac{d\boldsymbol{q^\tau}}{dt} - H(\boldsymbol{p^\tau}, \boldsymbol{q^\tau}) \right) \mathrm{d}t - \sum_{n=-1}^{N-1} [[\boldsymbol{q^\tau}]] \{\{\boldsymbol{p^\tau}\}\}_\alpha^\beta |_{t^{n+1}} \right\}, \tag{3.38}$$

where $[[.]]$ and $\{\{.\}\}_\alpha^\beta$ denote the jump and average operators, with $\alpha + \beta = 1$ and $\alpha, \beta > 0$. Gagarina *et al.* [53] showed that through the choice of the order of the polynomials $\boldsymbol{p^\tau}$ and $\boldsymbol{q^\tau}$, of the quadrature rule and of the weights α and β, various stable temporal schemes may be obtained for the discretisation of (3.38).

In our case, the Hamiltonian $\tilde{H}$ admits an additional explicit time dependency through the term $W(x, y, t) = L_w - \tilde{R}(x, y, t)$. Bokhove and Kalogirou [20] showed that non-autonomous systems of the form of (3.35) may be expressed as autonomous systems by introducing a new time coordinate τ and its conjugate p, such that $t = t(\tau)$ is an auxiliary variable, with $dt/d\tau = 1$ and $t(0) = 0s$. They transformed the Hamiltonian $H(\boldsymbol{p_1}, \boldsymbol{h}, t)$ into a so-called "Kamiltonian" system, defined as $K(\boldsymbol{p_1}, \boldsymbol{h}, t, p) = H(\boldsymbol{p_1}, \boldsymbol{h}, t) + p$ for which the energy is conserved and the equations of the form (3.36) are recovered for the variables $\boldsymbol{P} = (\boldsymbol{p_1}, p)$ and $\boldsymbol{Q} = (\boldsymbol{h}, t)$. In the Hamiltonian $\tilde{H}$, the explicit time dependency W is also involved in our momentum variable p_{1k}, since

$$p_{1k} \quad = \quad M_{kq}\psi_{1q} \quad = \quad \int_{\hat{\Omega}_{x,y}} \left[W(x,y,t)\varphi_k(x,y)\varphi_q(x,y)\right]\psi_{1q}(t)\,\mathrm{d}x\,\mathrm{d}y. \tag{3.39}$$

A logical choice is thus to define a new momentum $\tilde{\boldsymbol{P}}$ as $\tilde{\boldsymbol{P}} = (\boldsymbol{p_1}, t)$, and introduce the conjugate q of t to get the corresponding coordinate $\tilde{\boldsymbol{Q}} = (\boldsymbol{h}, q)$. In order to obtain equations of the form (3.36) in terms of $\tilde{\boldsymbol{P}}$ and $\tilde{\boldsymbol{Q}}$, we define an adjoint Kamiltonian as

$$\tilde{K}(\tilde{\boldsymbol{P}}, \tilde{\boldsymbol{Q}}) := \tilde{K}(\boldsymbol{p_1}, \boldsymbol{h}, t, q) = \tilde{H}(\boldsymbol{p_1}, \boldsymbol{h}, t) - q. \tag{3.40}$$

Through the definition of $t(\tau)$, we have

$$\frac{d\boldsymbol{h}}{dt} = \frac{d\boldsymbol{h}}{dt}\frac{dt}{d\tau} = \frac{d\boldsymbol{h}}{d\tau}, \tag{3.41a}$$

$$\frac{d\boldsymbol{p_1}}{dt} = \frac{d\boldsymbol{p_1}}{dt}\frac{dt}{d\tau} = \frac{d\boldsymbol{p_1}}{d\tau}, \tag{3.41b}$$

using which the variational principle (3.35) becomes

$$\begin{aligned} 0 =& \delta \int_0^T \left[\boldsymbol{p_1}\frac{d\boldsymbol{h}}{d\tau} - q\frac{dt}{d\tau} - \tilde{K}(\boldsymbol{p_1}, \boldsymbol{h}, t, q)\right] \mathrm{d}\tau \\ =& \int_0^T \Bigg[\left(\frac{d\boldsymbol{h}}{d\tau} - \frac{\partial\tilde{K}}{\partial\boldsymbol{p_1}}\right)\delta\boldsymbol{p_1} - \left(\frac{d\boldsymbol{p_1}}{d\tau} + \frac{\partial\tilde{K}}{\partial\boldsymbol{h}}\right)\delta\boldsymbol{h} \\ & - \left(\frac{dt}{d\tau} + \frac{\partial\tilde{K}}{\partial q}\right)\delta q + \left(\frac{dq}{d\tau} - \frac{\partial\tilde{K}}{\partial t}\right)\delta t \Bigg] \mathrm{d}\tau, \end{aligned} \tag{3.42}$$

where we have used that $\delta\boldsymbol{h}(0) = \delta\boldsymbol{h}(T) = 0$ and $\delta t(0) = \delta t(T) = 0$. Arbitrariness of $\delta\boldsymbol{h}$, $\delta\boldsymbol{p_1}$, δt and δq leads to the following equations:

$$\frac{d\boldsymbol{h}}{d\tau} = \frac{\partial\tilde{K}}{\partial\boldsymbol{p_1}}, \qquad \frac{d\boldsymbol{p_1}}{d\tau} = -\frac{\partial\tilde{K}}{\partial\boldsymbol{h}}, \qquad \frac{dt}{d\tau} = -\frac{\partial\tilde{K}}{\partial q}, \qquad \frac{dq}{d\tau} = \frac{\partial\tilde{K}}{\partial t}. \tag{3.43}$$

Equations. (3.43) may be combined to take the form of (3.36):

$$\frac{d(\boldsymbol{h}, q)}{d\tau} = \frac{\partial\tilde{K}}{\partial(\boldsymbol{p_1}, t)} \qquad \text{and} \qquad \frac{d(\boldsymbol{p_1}, t)}{d\tau} = -\frac{\partial\tilde{K}}{\partial(\boldsymbol{h}, q)} \tag{3.44}$$

$$\Rightarrow \quad \frac{d\tilde{\boldsymbol{Q}}}{d\tau} = \frac{\partial \tilde{K}}{\partial \tilde{\boldsymbol{P}}} \qquad \text{and} \qquad \frac{d\tilde{\boldsymbol{P}}}{d\tau} = -\frac{\partial \tilde{K}}{\partial \tilde{\boldsymbol{Q}}}. \tag{3.45}$$

From Eqs. (3.41) and (3.40), Eqs. (3.43) are also equivalent to

$$\frac{d\boldsymbol{h}}{dt} = \frac{\partial \tilde{H}}{\partial \boldsymbol{p_1}}, \qquad \frac{d\boldsymbol{p_1}}{dt} = -\frac{\partial \tilde{H}}{\partial \boldsymbol{h}}, \qquad \frac{dt}{d\tau} = 1, \qquad \frac{dq}{d\tau} = \frac{\partial \tilde{H}}{\partial t}, \tag{3.46}$$

in which we recover Eqs. (3.36). The temporal schemes introduced by Gagarina *et al.* [53] for autonomous systems may be applied to our non-autonomous system as long as the explicit time dependence $W(x, y, t)$ is evaluated as our momentum variable $\boldsymbol{p_1}$. By setting $\boldsymbol{p} = \boldsymbol{p_1}$, $\boldsymbol{q} = \boldsymbol{h}$, and $H(\boldsymbol{p}, \boldsymbol{q}) = \tilde{H}(\boldsymbol{p_1}, \boldsymbol{h}, t)$ in Eq. (3.38), the following discrete variational principle is obtained

$$0 = \delta \left\{ \sum_{n=0}^{N-1} \int_{t^n}^{t^{n+1}} \left(\boldsymbol{p_1^\tau} \frac{d\boldsymbol{h^\tau}}{dt} - \tilde{H}(\boldsymbol{p_1^\tau}, \boldsymbol{h^\tau}, t) \right) \mathrm{d}t - \sum_{n=-1}^{N-1} [[\boldsymbol{h^\tau}]] \{\{\boldsymbol{p_1^\tau}\}\}_\alpha^\beta |_{t^{n+1}} \right\}. \tag{3.47}$$

Two examples of resulting temporal schemes are now presented; both are used in our simulations.

3.4.2 1st–order symplectic-Euler scheme

We first apply the 1st–order symplectic-Euler scheme to Eq. (3.47). To obtain a 1st–order scheme, we approximate $\boldsymbol{p_1}$ and $\boldsymbol{h}$ with continuous constant basis functions $\boldsymbol{p_1^\tau}$ and $\boldsymbol{h^\tau}$ on each interval $[t^n, t^{n+1}]$, but discontinuous across the interface t^n, with left and right values

$$(\boldsymbol{h^{n,-}}, \boldsymbol{p_1^{n,-}}) \qquad \text{and} \qquad (\boldsymbol{h^{n,+}}, \boldsymbol{p_1^{n,+}}), \tag{3.48}$$

for $n \in [0, N-1]$. As $\boldsymbol{h^\tau}$ is constant on each interval, we have

$$\int_{t^n}^{t^{n+1}} \boldsymbol{p_1^\tau} \frac{d\boldsymbol{h^\tau}}{dt} \, \mathrm{d}t = 0 \quad \forall n \in [0, N-1], \tag{3.49}$$

simplifying Eq. (3.47) to

$$0 = \delta \left\{ \sum_{n=0}^{N-1} \int_{t^n}^{t^{n+1}} -\tilde{H}(\boldsymbol{p_1^\tau}, \boldsymbol{h^\tau}, t) \, \mathrm{d}t - \sum_{n=-1}^{N-1} [[\boldsymbol{h^\tau}]] \{\{\boldsymbol{p_1^\tau}\}\}_\alpha^\beta |_{t^{n+1}} \right\}. \tag{3.50}$$

Following [53] and the conclusions from our Kamiltonian system (3.43), we set $(\boldsymbol{p}_1^{\boldsymbol{\tau}}, \boldsymbol{h}^{\boldsymbol{\tau}}, t) = (\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n,+}, t^{n,+}) = (\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+})$, $\alpha = 0$, $\beta = 1$, and use the quadrature rule

$$\int_{t^n}^{t^{n+1}} \tilde{H}(\boldsymbol{p}_1^{\boldsymbol{\tau}}, \boldsymbol{h}^{\boldsymbol{\tau}}, t)\,\mathrm{d}t = \Delta t \tilde{H}(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}). \tag{3.51}$$

As a consequence, the discrete variational principle becomes

$$0 = \delta \left\{ \sum_{n=0}^{N-1} \Delta t \tilde{H}\left(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}\right) + \sum_{n=-1}^{N-1} \left(\boldsymbol{h}^{n+1,-} - \boldsymbol{h}^{n+1,+}\right) \boldsymbol{p}_1^{n+1,+} \right\}. \tag{3.52}$$

Variations of (3.52) with respect to $\boldsymbol{p}^{\boldsymbol{\tau}} = \boldsymbol{p}_1^{n,+}$ and $\boldsymbol{h}^{\boldsymbol{\tau}} = \boldsymbol{h}^{n+1,-}$ lead to

$$\begin{aligned} 0 = &\sum_{n=0}^{N-1} \left[\Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}\right)}{\partial \boldsymbol{p}_1^{n,+}} \delta \boldsymbol{p}_1^{n,+} + \Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}\right)}{\partial \boldsymbol{h}^{n+1,-}} \delta \boldsymbol{h}^{n+1,-} \right] \\ &+ \sum_{n=-1}^{N-1} \left[\left(\boldsymbol{h}^{n,-} - \boldsymbol{h}^{n,+}\right) \delta \boldsymbol{p}_1^{n,+} + \left(\boldsymbol{p}_1^{n+1,+} - \boldsymbol{p}_1^{n,+}\right) \delta \boldsymbol{h}^{n+1,-} \right]. \end{aligned} \tag{3.53}$$

Arbitrariness of $\delta \boldsymbol{p}_1^{n,+}$ and $\delta \boldsymbol{h}^{n+1,-}$ then yields

$$\delta \boldsymbol{p}_1^{n,+}: \quad \boldsymbol{h}^{n,+} = \boldsymbol{h}^{n,-} + \Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}\right)}{\partial \boldsymbol{p}_1^{n,+}}, \tag{3.54a}$$

$$\delta \boldsymbol{h}^{n+1,-}: \quad \boldsymbol{p}_1^{n+1,+} = \boldsymbol{p}_1^{n,+} - \Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^{n,+}, \boldsymbol{h}^{n+1,-}, t^{n,+}\right)}{\partial \boldsymbol{h}^{n+1,-}}. \tag{3.54b}$$

By construction, $\boldsymbol{h}^{n,+} = \boldsymbol{h}^{n+1,-} := \boldsymbol{h}^{n+1}$, as $\boldsymbol{h}^{\tau}$ is constant over a cell. Similarly, we denote $\boldsymbol{p}_1^n := \boldsymbol{p}_1^{n,+}$, [53], and $t^n := t^{n,+}$. We therefore obtain the symplectic-Euler scheme for non-autonomous Hamiltonian system as

$$\boldsymbol{h}^{n+1} = \boldsymbol{h}^n + \Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^n, \boldsymbol{h}^{n+1}, t^n\right)}{\partial \boldsymbol{p}_1^n}, \tag{3.55a}$$

$$\boldsymbol{p}_1^{n+1} = \boldsymbol{p}_1^n - \Delta t \frac{\partial \tilde{H}\left(\boldsymbol{p}_1^n, \boldsymbol{h}^{n+1}, t^n\right)}{\partial \boldsymbol{h}^{n+1}}. \tag{3.55b}$$

From our conclusions in section 3.4.1, solving Eq. (3.55) is equivalent to solving

$$\begin{cases} \boldsymbol{h}^{n+1} = \boldsymbol{h}^{n} + \Delta t \dfrac{\partial H\left(\boldsymbol{p}_{\mathbf{1}}^{n}, \boldsymbol{h}^{n+1}, \boldsymbol{\psi}_{\boldsymbol{i'}}^{*}, t^{n}\right)}{\partial \boldsymbol{p}_{\mathbf{1}}^{n}} \\ \dfrac{\partial H\left(\boldsymbol{p}_{\mathbf{1}}^{n}, \boldsymbol{h}^{n+1}, \boldsymbol{\psi}_{\boldsymbol{i'}}^{*}, t^{n}\right)}{\partial \boldsymbol{\psi}_{\boldsymbol{i'}}^{*}} = 0, \qquad \text{for each } i' \in [2, n_z+1], \end{cases} \tag{3.56a}$$

$$\boldsymbol{p}_{\mathbf{1}}^{n+1} = \boldsymbol{p}_{\mathbf{1}}^{n} - \Delta t \frac{\partial H\left(\boldsymbol{p}_{\mathbf{1}}^{n}, \boldsymbol{h}^{n+1}, \boldsymbol{\psi}_{\boldsymbol{i'}}^{*} t^{n}\right)}{\partial \boldsymbol{h}^{n+1}}, \tag{3.56b}$$

where the second Equation in (3.56a) enables us to obtain $\boldsymbol{\psi}_{\boldsymbol{i'}}$ in terms of $\boldsymbol{h}^{n+1}$, $\boldsymbol{\psi}_{\mathbf{1}}^{n}$ and t^n. Note that Eq. (3.56b) is solved on its own since $\boldsymbol{\psi}_{\boldsymbol{i'}}^{*} = \boldsymbol{\psi}_{\boldsymbol{i'}}(\boldsymbol{h}^{n+1}, \boldsymbol{\psi}_{\mathbf{1}}^{n}, t^{n})$ is known from step (3.56a).

Substituting the Hamiltonian Eq. (3.32) into Eq. (3.56a) leads to the first step of the symplectic-Euler scheme in fully discrete form (*cf.* Appendix B.2, Eqs. B.18-B.19). This implicit step updates both $\boldsymbol{h}$ at time t^{n+1} and $\boldsymbol{\psi}_{\boldsymbol{i'}}$, for $i' \in [2, N_z]$ at an auxiliary time t^* corresponding to $\boldsymbol{\psi}_{\boldsymbol{i'}}^{*} = \psi_{i'}(\boldsymbol{h}^{n+1}, \boldsymbol{p}_{\mathbf{1}}^{n}, t^{n})$. It is solved in a mixed system, as explained in the tutorial Chapter 6. The second step of the symplectic-Euler scheme, which updates $p_{1k} = M_{kq}^{-1}\psi_{1q}$ at time t^{n+1}, is obtained in Eq. B.20 and is explicit for $\boldsymbol{p}_{\mathbf{1}}$. For analysis purpose, such as visualising the velocity potential in the full 3D domain, one may also update $\hat{\psi}$ at time t^{n+1} by solving Eq. (3.34) with h, ψ_1 and t evaluated at time t^{n+1}. The linear equation updating $\hat{\psi}^{n+1}$ is given in Appendix B.2.

3.4.3 Second-order Störmer-Verlet scheme

In order to obtain a second-order scheme, the polynomials $\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}$ and $\boldsymbol{h}^{\boldsymbol{\tau}}$ must be expanded with linear basis functions. As advised in [53], we approximate $\boldsymbol{h}^{\boldsymbol{\tau}}$ in terms of $\boldsymbol{h}$ at times $t^{n,+}$ and $t^{n+1,-}$ (trapezoidal rule), while $\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}$ is approximated in terms of $\boldsymbol{p}_{\mathbf{1}}^{n,+}$ and $\boldsymbol{p}_{\mathbf{1}}^{n+1/2}$ (mid-point rule). We therefore define $\boldsymbol{h}^{\boldsymbol{\tau}}$ and $\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}$ so that they satisfy:

$$\boldsymbol{h}^{\boldsymbol{\tau}}(t^{n,+}) = \boldsymbol{h}^{\boldsymbol{n},+}, \tag{3.57a}$$

$$\boldsymbol{h}^{\boldsymbol{\tau}}(t^{n+1,-}) = \boldsymbol{h}^{\boldsymbol{n+1},-}, \tag{3.57b}$$

$$\boldsymbol{h}^{\boldsymbol{\tau}}(t^{n+1/2}) = \frac{1}{2}(\boldsymbol{h}^{\boldsymbol{n},+} + \boldsymbol{h}^{\boldsymbol{n+1},-}), \tag{3.57c}$$

$$\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}(t^{n,+}) = \boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{n},+}, \tag{3.57d}$$

$$\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}(t^{n+1,-}) = 2\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{n+1/2}} - \boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{n},+}, \tag{3.57e}$$

$$\boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{\tau}}(t^{n+1/2}) = \boldsymbol{p}_{\mathbf{1}}^{\boldsymbol{n+1/2}}. \tag{3.57f}$$

The following expansions, introduced by [53], satisfy the requirements (3.57):

$$\boldsymbol{h}^{\boldsymbol{\tau}}(t) = \frac{t^{n+1}-t}{\Delta t}\boldsymbol{h}^{n,+} + \frac{t-t^n}{\Delta t}\boldsymbol{h}^{n+1,-}, \tag{3.58a}$$

$$\boldsymbol{p_1^{\tau}}(t) = \frac{2(t-t^n)}{\Delta t}\boldsymbol{p_1}^{n+1/2} + \frac{t^n+t^{n+1}-2t}{\Delta t}\boldsymbol{p_1}^{n,+}. \tag{3.58b}$$

The integral of the Hamiltonian may also be approximated in terms of $\boldsymbol{p_1}^{n+1/2}$, $\boldsymbol{h^{n,+}}$ and $\boldsymbol{h^{n+1,-}}$ as [53]:

$$\int_{t^n}^{t^{n+1}} \tilde{H}(\boldsymbol{h^{\tau}}, \boldsymbol{p_1^{\tau}}, t)\,\mathrm{d}t = \frac{\Delta t}{2}\Big[\tilde{H}(\boldsymbol{h^{n,+}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}) + \tilde{H}(\boldsymbol{h^{n+1,-}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})\Big]. \tag{3.59}$$

Substitution of Eqs. (3.58) and (3.59) into the discrete variational principle (3.47) leads to

$$\begin{aligned}\delta\mathcal{L}(\boldsymbol{h^{\tau}}, \boldsymbol{p_1^{\tau}}, t) = \delta\sum_{n=0}^{N-1}\Bigg[&\boldsymbol{p_1}^{n+1/2}\left(\boldsymbol{h}^{n+1,-}-\boldsymbol{h}^{n,+}\right)\\ &-\frac{\Delta t}{2}\Big(\tilde{H}(\boldsymbol{h^{n,+}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}) + \tilde{H}(\boldsymbol{h^{n+1,-}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})\Big)\Bigg]\\ &-\delta\sum_{n=-1}^{N-1}\Big(2\alpha\boldsymbol{p_1}^{n+1/2} - \alpha\boldsymbol{p_1}^{n,+} + \beta\boldsymbol{p_1}^{n+1,+}\Big)\left(\boldsymbol{h}^{n+1,-}-\boldsymbol{h}^{n+1,+}\right) = 0.\end{aligned} \tag{3.60}$$

The variations with respect to $\boldsymbol{p_1}^{n+1/2}$, $\boldsymbol{p_1}^{n,+}$, $\boldsymbol{h}^{n,+}$ and $\boldsymbol{h}^{n+1,-}$, with end-point conditions $\delta\boldsymbol{h}^{0,-} = 0$, $\delta\boldsymbol{p_1}^{0,-} = 0$, $\delta\boldsymbol{h}^{N,+} = 0$ and $\delta\boldsymbol{p_1}^{N,+} = 0$ yield the following system of equations:

$$\delta\boldsymbol{p_1}^{n,+} : \alpha(\boldsymbol{h}^{n+1,-}-\boldsymbol{h}^{n+1,+}) = \beta(\boldsymbol{h}^{n,-}-\boldsymbol{h}^{n,+}), \tag{3.61a}$$

$$\delta\boldsymbol{h}^{n,+} : \boldsymbol{p_1}^{n+1/2} = \alpha(2\boldsymbol{p_1}^{n-1/2} - \boldsymbol{p_1}^{n-1,+}) + \beta\boldsymbol{p_1}^{n,+} - \frac{\Delta t}{2}\frac{\partial\tilde{H}\left(\boldsymbol{h}^{n,+}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}\right)}{\partial\boldsymbol{h}^{n,+}}, \tag{3.61b}$$

$$\begin{aligned}\delta\boldsymbol{p_1}^{n+1/2} : 0 = &(1-2\alpha)\boldsymbol{h}^{n+1,-} + 2\alpha\boldsymbol{h}^{n+1,+} - \boldsymbol{h}^{n,+}\\ &-\frac{\Delta t}{2}\Big(\frac{\partial\tilde{H}(\boldsymbol{h^{n,+}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})}{\partial\boldsymbol{p_1}^{n+1/2}} + \frac{\partial\tilde{H}(\boldsymbol{h^{n+1,-}}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})}{\partial\boldsymbol{p_1}^{n+1/2}}\Big),\end{aligned} \tag{3.61c}$$

$$\delta\boldsymbol{h}^{n+1,-} : \beta\boldsymbol{p_1}^{n+1,+} = (1-2\alpha)\boldsymbol{p_1}^{n+1/2} + \alpha\boldsymbol{p_1}^{n,+} - \frac{\Delta t}{2}\frac{\partial\tilde{H}\left(\boldsymbol{h}^{n+1,-}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}\right)}{\partial\boldsymbol{h}^{n+1,-}}. \tag{3.61d}$$

In section 3.4.4, we follow the method described by [53] to show that the scheme (3.61) is stable for $\alpha \in [0, 0.5]$ and $|\omega\Delta t| \leq 2$ (with ω the maximum frequency of the discrete waves). Consequently, setting $\alpha = 0$ and $\beta = 1$ in the scheme (3.61) leads to:

$$\delta \boldsymbol{p}_{\mathbf{1}}^{n,+}: \quad \boldsymbol{h}^{n,-} = \boldsymbol{h}^{n,+}, \tag{3.62a}$$

$$\delta \boldsymbol{h}^{n,+}: \quad \boldsymbol{p}_{\mathbf{1}}^{n+1/2} = \boldsymbol{p}_{\mathbf{1}}^{n,+} - \frac{\Delta t}{2} \frac{\partial \tilde{H}\left(\boldsymbol{h}^{n,+}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n,+}}, \tag{3.62b}$$

$$\begin{aligned}\delta \boldsymbol{p}_{\mathbf{1}}^{n+1/2}: 0 = \boldsymbol{h}^{n+1,-} - \boldsymbol{h}^{n,+} - \frac{\Delta t}{2}\Big(& \frac{\partial \tilde{H}(\boldsymbol{h}^{n,+}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p}_{\mathbf{1}}^{n+1/2}} \\ & + \frac{\partial \tilde{H}(\boldsymbol{h}^{n+1,-}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p}_{\mathbf{1}}^{n+1/2}}\Big),\end{aligned} \tag{3.62c}$$

$$\delta \boldsymbol{h}^{n+1,-}: \quad \boldsymbol{p}_{\mathbf{1}}^{n+1,+} = \boldsymbol{p}_{\mathbf{1}}^{n+1/2} - \frac{\Delta t}{2} \frac{\partial \tilde{H}\left(\boldsymbol{h}^{n+1,-}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n+1,-}}. \tag{3.62d}$$

Equation (3.62) ensures continuous depth at the time interfaces, so we can set $\boldsymbol{h}^{n,+} = \boldsymbol{h}^{n,-} := \boldsymbol{h}^n$. Moreover, we set $\boldsymbol{p}_{\mathbf{1}}^n = \{\{\boldsymbol{p}_{\mathbf{1}}\}\}_{\alpha}^{\beta}|_{t^n} = \boldsymbol{p}_{\mathbf{1}}^{n,+}$. Substituting these notations into (3.62) results in the adjoint Störmer-Verlet scheme:

$$\delta \boldsymbol{p}_{\mathbf{1}}^{n}: \quad [[\boldsymbol{h}]]_{t^n} = 0, \tag{3.63a}$$

$$\delta \boldsymbol{h}^{n}: \quad \boldsymbol{p}_{\mathbf{1}}^{n+1/2} = \boldsymbol{p}_{\mathbf{1}}^{n} - \frac{\Delta t}{2} \frac{\partial \tilde{H}\left(\boldsymbol{h}^{n}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n}}, \tag{3.63b}$$

$$\begin{aligned}\delta \boldsymbol{p}_{\mathbf{1}}^{n+1/2}: 0 = \boldsymbol{h}^{n+1} - \boldsymbol{h}^{n} - \frac{\Delta t}{2}\Big(& \frac{\partial \tilde{H}(\boldsymbol{h}^{n}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p}_{\mathbf{1}}^{n+1/2}} \\ & + \frac{\partial \tilde{H}(\boldsymbol{h}^{n+1}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p}_{\mathbf{1}}^{n+1/2}}\Big),\end{aligned} \tag{3.63c}$$

$$\delta \boldsymbol{h}^{n+1}: \quad \boldsymbol{p}_{\mathbf{1}}^{n+1} = \boldsymbol{p}_{\mathbf{1}}^{n+1/2} - \frac{\Delta t}{2} \frac{\partial \tilde{H}\left(\boldsymbol{h}^{n+1}, \boldsymbol{p}_{\mathbf{1}}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n+1}}. \tag{3.63d}$$

Note that this derivation yields the adjoint Störmer-Verlet scheme, in which the momentum is evaluated at the intermediate time $t^{n+1/2}$. We have made that choice rather than the usual Störmer-Verlet scheme (as presented in detail in [53]) as this enables the evaluation of the wavemaker terms at only $t^{n+1/2}$ in every steps. If one uses the usual Störmer-Verlet scheme instead, in which $\boldsymbol{h}$ and $\boldsymbol{p}_{\mathbf{1}}$ are reversed, then the wavemaker terms need to be evaluated at times t^n and t^{n+1} in the second step (as would be done for $\boldsymbol{p}_{\mathbf{1}}$).

As explained in section 3.4.1, solving Eqs. (3.63) is equivalent to solving:

$$\begin{cases} 0 = \boldsymbol{p_1}^{n+1/2} - \boldsymbol{p_1}^n + \dfrac{\Delta t}{2}\dfrac{\partial H\left(\boldsymbol{h}^n, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^*, t^{n+1/2}\right)}{\partial \boldsymbol{h}^n}, \\ 0 = \dfrac{\partial H\left(\boldsymbol{h}^n, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^*, t^{n+1/2}\right)}{\partial \psi_{i'}^*}, \end{cases} \tag{3.64a}$$

$$\begin{cases} 0 = \boldsymbol{h}^{n+1} - \boldsymbol{h}^n - \dfrac{\Delta t}{2}\Big(\dfrac{\partial H(\boldsymbol{h^n}, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^*, t^{n+1/2})}{\partial \boldsymbol{p_1}^{n+1/2}} \\ \qquad\qquad + \dfrac{\partial H(\boldsymbol{h^{n+1}}, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^{**}, t^{n+1/2})}{\partial \boldsymbol{p_1}^{n+1/2}}\Big), \\ 0 = \dfrac{\partial H(\boldsymbol{h^{n+1}}, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^{**}, t^{n+1/2})}{\partial \psi_{i'}^{**}}, \end{cases} \tag{3.64b}$$

$$\boldsymbol{p_1}^{n+1} = \boldsymbol{p_1}^{n+1/2} - \frac{\Delta t}{2}\frac{\partial H\left(\boldsymbol{h}^{n+1}, \boldsymbol{p_1}^{n+1/2}, \psi_{i'}^{**}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n+1}}, \tag{3.64c}$$

where Eq. (3.64a) aims to eliminate $\psi_{i'}^* = \psi_{i'}(\boldsymbol{h}^n, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})$ while updating $\boldsymbol{p_1}$ at $t^{n+1/2}$, and Eq. (3.64b) aims to eliminate $\psi_{i'}^{**} = \psi_{i'}(\boldsymbol{h}^{n+1}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})$ while updating $\boldsymbol{h}$ at t^{n+1}. Note that Eq. (3.64) is fully explicit, so $\psi_{i'}$ does not need to be updated simultaneously. The detailed expression of each of these weak formulations is given in Appendix B.3.

3.4.4 Stability of the schemes

To ensure stability of the temporal schemes, the time step must be limited to certain values. To estimate this supremum, a Fourier analysis is conducted for each scheme.

Stability of the 1^{st}–order Symplectic-Euler scheme

To study the stability of the symplectic-Euler scheme, we consider the energy of an harmonic oscillator, given by

$$\tilde{H}(\boldsymbol{p_1}, \boldsymbol{h}) = \frac{1}{2}\boldsymbol{p_1}^2 + \frac{1}{2}\omega^2\boldsymbol{h}^2, \tag{3.65}$$

where ω is the frequency of the oscillator. Substitution of this Hamiltonian into the symplectic-Euler scheme (3.55) leads to the following system of equations

$$\boldsymbol{h}^{n+1} = \boldsymbol{h}^n + \Delta t \boldsymbol{p}_{\boldsymbol{1}}^n, \tag{3.66a}$$

$$\boldsymbol{p}_{\boldsymbol{1}}^{n+1} = \boldsymbol{p}_{\boldsymbol{1}}^n - \omega^2 \Delta t \boldsymbol{h}^{n+1}. \tag{3.66b}$$

Substituting the Ansatz functions $\boldsymbol{p}_{\boldsymbol{1}}^n = p_1\lambda^n$ and $\boldsymbol{h}^n = h\lambda^n$ leads to

$$h\lambda^{n+1} = h\lambda^n + \Delta t p_1 \lambda^n, \tag{3.67a}$$

$$p_1\lambda^{n+1} = p_1\lambda^n - \omega^2 \Delta t h \lambda^{n+1}. \tag{3.67b}$$

Division by λ^n reduces the above system of equations to

$$(1-\lambda)h = -\Delta t p_1, \tag{3.68a}$$

$$(1-\lambda)p_1 = \omega^2 \Delta t h \lambda, \tag{3.68b}$$

which in matrix form reads

$$\begin{pmatrix} (1-\lambda) & \Delta t \\ -\omega^2\Delta t\lambda & (1-\lambda) \end{pmatrix} \begin{pmatrix} h \\ p_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{3.69}$$

The corresponding characteristic polynomial is

$$\lambda^2 + (\omega^2\Delta t^2 - 2)\lambda + 1 = 0. \tag{3.70}$$

A solution with coefficient $\boldsymbol{p}_{\boldsymbol{1}}^n = p_1\lambda^n$ or $\boldsymbol{h}^n = h\lambda^n$ can only be stable if $|\lambda| < 1$, as otherwise the solution would blow-up with time (that is, as $n \to \infty$). In Appendix B.4, we show that $|\lambda| < 1$ if and only if

$$\Delta t \leq \frac{2}{\omega}, \tag{3.71}$$

which is thus the time-step restriction ensuring stability of the symplectic-Euler scheme.

Stability of the second-order Störmer-Verlet scheme

The stability condition for the Störmer-Verlet scheme is obtained by following the method used in [53]. First, the system of equations (3.61) is reformulated in terms of the unknowns $[[\boldsymbol{p}_{\boldsymbol{1}}]]|_{t_{n+1}}$,

$\boldsymbol{p_1}^{n+1/2}$, $[[\boldsymbol{h}]]|_{t_{n+1}}$ and $\boldsymbol{h}^{n+1,-}$:

$$\alpha[[\boldsymbol{h}]]|_{t_{n+1}} = \beta[[\boldsymbol{h}]]|_{t_n}, \tag{3.72a}$$

$$\boldsymbol{p_1}^{n+1/2} = \boldsymbol{p_1}^{n,+} + \alpha[[\boldsymbol{p_1}]]|_{t_n} - \frac{\Delta t}{2}\frac{\partial \tilde{H}\left(\boldsymbol{h}^{n,+}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n,+}}, \tag{3.72b}$$

$$\begin{aligned}\boldsymbol{h}^{n+1,-} =& \boldsymbol{h}^{n,+} + 2\alpha[[\boldsymbol{h}]]|_{t_{n+1}} + \frac{\Delta t}{2}\Big(\frac{\partial \tilde{H}(\boldsymbol{h}^{n,+}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p_1}^{n+1/2}} \\ &+ \frac{\partial \tilde{H}(\boldsymbol{h}^{n+1,-}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2})}{\partial \boldsymbol{p_1}^{n+1/2}}\Big),\end{aligned} \tag{3.72c}$$

$$\beta[[\boldsymbol{p_1}]]|_{t_{n+1}} = \boldsymbol{p_1}^{n+1/2} - \boldsymbol{p_1}^{n,+} + \frac{\Delta t}{2}\frac{\partial \tilde{H}\left(\boldsymbol{h}^{n+1,-}, \boldsymbol{p_1}^{n+1/2}, t^{n+1/2}\right)}{\partial \boldsymbol{h}^{n+1,-}}, \tag{3.72d}$$

where we used the definition of the jump $[[.]]|_{t_{n+1}}$, that is

$$\boldsymbol{p_1}^{n+1,+} = 2\boldsymbol{p_1}^{n+1/2} - \boldsymbol{p_1}^{n,+} - [[\boldsymbol{p_1}]]|_{t_{n+1}}, \tag{3.73a}$$

$$\boldsymbol{h}^{n+1,+} = \boldsymbol{h}^{n+1,-} - [[\boldsymbol{h}]]|_{t_{n+1}}. \tag{3.73b}$$

Substitution of the Hamiltonian (3.65) into the system of equations (3.72) leads to

$$\alpha[[\boldsymbol{h}]]|_{t_{n+1}} = \beta[[\boldsymbol{h}]]|_{t_n}, \tag{3.74a}$$

$$\boldsymbol{p_1}^{n+1/2} = \boldsymbol{p_1}^{n,+} + \alpha[[\boldsymbol{p_1}]]|_{t_n} - \frac{\Delta t}{2}\omega^2 \boldsymbol{h}^{n,+}, \tag{3.74b}$$

$$\boldsymbol{h}^{n+1,-} = \boldsymbol{h}^{n,+} + 2\alpha[[\boldsymbol{h}]]|_{t_{n+1}} + \Delta t \boldsymbol{p_1}^{n+1/2}, \tag{3.74c}$$

$$\beta[[\boldsymbol{p_1}]]|_{t_{n+1}} = \boldsymbol{p_1}^{n+1/2} - \boldsymbol{p_1}^{n,+} + \frac{\Delta t}{2}\omega^2 \boldsymbol{h}^{n+1,-}. \tag{3.74d}$$

The above system of equations may be written in matrix form as

$$\begin{pmatrix} [[\boldsymbol{h}]]|_{t_{n+1}} \\ \boldsymbol{h}^{n+1,+} \\ [[\boldsymbol{p_1}]]|_{t_{n+1}} \\ \boldsymbol{p_1}^{n+1,+} \end{pmatrix} = C \begin{pmatrix} [[\boldsymbol{h}]]|_{t_n} \\ \boldsymbol{h}^{n,+} \\ [[\boldsymbol{p_1}]]|_{t_n} \\ \boldsymbol{p_1}^{n,+} \end{pmatrix}, \tag{3.75}$$

with C the matrix

$$\begin{pmatrix} \frac{\beta}{\alpha} & 0 & 0 & 0 \\ \beta(2-\frac{1}{\alpha}) & (1-\frac{\omega^2\Delta t^2}{2}) & \alpha\Delta t & \Delta t \\ \omega^2\Delta t & -\frac{\omega^4\Delta t^3}{4\beta} & \frac{\alpha}{\beta}\left(1+\frac{\omega^2\Delta t^2}{2}\right) & \frac{\omega^2\Delta t^2}{2\beta} \\ -\omega^2\Delta t & \frac{\omega^4\Delta t^3}{4\beta}-\omega^2\Delta t & 2\alpha-\frac{\alpha}{\beta}\left(1+\frac{\omega^2\Delta t^2}{2}\right) & 1-\frac{\omega^2\Delta t^2}{2\beta} \end{pmatrix}. \tag{3.76}$$

Assuming a continuous water depth across the time cells, that is $[[\boldsymbol{h}]]_{t_n} = 0$, the first row and column of the matrix C may be removed, to lead to

$$\begin{pmatrix} (1-\frac{\omega^2\Delta t^2}{2}) & \alpha\Delta t & \Delta t \\ -\frac{\omega^4\Delta t^3}{4\beta} & \frac{\alpha}{\beta}\left(1+\frac{\omega^2\Delta t^2}{2}\right) & \frac{\omega^2\Delta t^2}{2\beta} \\ \frac{\omega^4\Delta t^3}{4\beta}-\omega^2\Delta t & 2\alpha-\frac{\alpha}{\beta}\left(1+\frac{\omega^2\Delta t^2}{2}\right) & 1-\frac{\omega^2\Delta t^2}{2\beta} \end{pmatrix}. \tag{3.77}$$

The characteristic polynomial of (3.77) is

$$\left(\frac{\alpha}{\beta}-\lambda\right)\left(1+\lambda\left(\omega^2\Delta t^2-2\right)+\lambda^2\right)=0, \tag{3.78}$$

and admits three solutions. The first solution is

$$\lambda=\frac{\alpha}{\beta}, \tag{3.79}$$

which satisfies the condition $|\lambda| \leq 1$ if and only if $\alpha \in [0, 0.5]$ and $\beta \in [0.5, 1]$ (with $\alpha + \beta = 1$). This condition justifies the choice of $\alpha = 0$ and $\beta = 1$ made in section 3.4.3 to obtain the adjoint Störmer-Verlet scheme (3.63). The other two solutions are obtained by solving

$$1+\left(\omega^2\Delta t^2-2\right)+\lambda^2=0, \tag{3.80}$$

which is the same polynomial as for the symplectic-Euler scheme in (3.70), for which the solutions satisfy the condition $|\lambda| \leq 1$ if and only if $\Delta t \leq 2/\omega$. Therefore, the stability condition for the adjoint Störmer-Verlet scheme obtained with $\alpha = 0$ and $\beta = 1$ is the same as for the symplectic-Euler scheme, that is

$$\Delta t \leq \frac{2}{\omega}. \tag{3.81}$$

The estimation of the time step from this stability condition is explained next.

Estimation of the time step

For both the symplectic-Euler and Störmer-Verlet schemes, stability is ensured as long as

$$\Delta t \leq \frac{2}{\omega}. \tag{3.82}$$

In order to estimate the largest time step that ensures stable simulations, the stability condition may be written

$$\Delta t \leq \frac{2}{\max(\omega)}. \tag{3.83}$$

From the linear dispersion relation

$$\omega = \sqrt{gk\tanh(kH_0)}, \tag{3.84}$$

so the maximal frequency is reached for short waves. Therefore, the stability condition (3.82) may be written as

$$\begin{aligned} \Delta t &\leq \frac{2}{\sqrt{g\max(k)\tanh(\max(k)H_0)}}, \\ \Rightarrow \Delta t &\leq \frac{2}{\sqrt{g\dfrac{2\pi}{\min(\lambda)}\tanh(\dfrac{2\pi}{\min(\lambda)}H_0)}}, \end{aligned} \tag{3.85}$$

using the definition of the wave number $k = \dfrac{2\pi}{\lambda}$. To be able to capture the wave, we must ensure that

$$\Delta x \leq \lambda, \tag{3.86}$$

where Δx is the spatial resolution and, for 1^{st}–order continuous expansions as used in our spatial discretisation, the minimal distance between two nodes. Therefore,

$$\min(\lambda) = \Delta x, \tag{3.87}$$

which leads to the time-step restriction

$$\Delta t \leq \frac{2}{\sqrt{g\dfrac{2\pi}{\Delta x}\tanh(\dfrac{2\pi}{\Delta x}H_0)}}. \tag{3.88}$$

In the next section, the implementation of the two above-mentioned schemes is explained.

3.5 Numerical solvers

3.5.1 Firedrake solvers

Firedrake [120, 9, 7, 71, 96] is used to solve the equations, for both the Symplectic-Euler (Eqs. (B.18) to (B.20)) and Störmer-Verlet (Eqs. (B.22) to (B.26)) time discretisations. Firedrake solves a nonlinear variational problem that takes the form

$$F(u; v) = 0, \tag{3.89}$$

for the unknown u, the test function v and the weak formulation F. This variational problem may be defined in Firedrake through the command

```
NL_problem = NonlinearVariationalProblem(F,u)
```

Similarly, a linear variational problem of the form

$$a(u; v) = L(v), \tag{3.90}$$

where a is bilinear and L is linear, may be defined numerically through the command

```
L_problem = LinearVariationalProblem(a,L,u)
```

The respective nonlinear and linear variational solvers are then defined as follows

```
# Nonlinear variational solver:
NL_solver = NonlinearVariationalSolver(NL_problem,solver_parameters={})
# Linear variational solver:
L_solver = LinearVariationalSolver(L_problem,solver_parameters={})
```

The *solver_parameters* argument enables to set options such as the factorisation method, the type of preconditioner, the convergence criteria, *etc*. In order to determine options required to optimise our solvers, a performance analysis is carried out in section 3.6.

3.5.2 Firedrake discretisation

The main advantage of using Firedrake is that the spatial discretisation is made internally. Therefore, the time-discrete space-continuous weak-formulations are implemented directly, by substituting the matrices (3.30) back into Eqs. (B.18), (B.19) and (B.20) for the Symplectic-Euler scheme and into Eqs. (B.22) to (B.26) for the Störmer-Verlet scheme. Moreover, the interior evaluations of the velocity potential may be written in vectorial form as

$$\psi_{i'}\tilde{\varphi}_{i'} = \hat{\psi}\hat{\varphi}^T, \tag{3.91}$$

with $\hat{\psi}$ and $\hat{\varphi}$ two vectors of dimension $(1, n_z)$ such that $\hat{\psi}(i) = \psi_{i+1}$ and $\hat{\varphi}(i) = \tilde{\varphi}_{i+1}$ for $i \in [1, n_z]$. Similarly, the z-discretised matrices (3.20) are split into four sub-matrices, constant in space and time, as follows:

$$X_{11} = X_{ij}[i=1, j=1], \tag{3.92a}$$

$$X_{1N} = X_{1j'} = X_{ij}[i=1, j=2:N_z], \tag{3.92b}$$

$$X_{N1} = X_{i'1} = X_{ij}[i=2:N_z, j=1], \tag{3.92c}$$

$$X_{NN} = X_{i'j'} = X_{ij}[i=2:N_z, j=2:N_z], \tag{3.92d}$$

where X denotes any matrix among $\tilde{A}$, $\tilde{M}$, $\tilde{D}$, $\tilde{S}$ and $\tilde{I}$. Then X_{11} is a scalar while X_{1N} and X_{N1} are respectively column and row vectors of dimension (n_z), and X_{NN} is a square matrix of dimension (n_z, n_z). Transforming the weak formulations (B.18) and (B.20) in terms of $h(x,y,t)$, $\psi_1(x,y,t)$ and $\psi_{i'}(x,y,t)$ and substituting (3.91) and (3.92) yield the space-continuous time-discrete weak formulations in vectorial form, as implemented with Firedrake. The resulting equations in Firedrake form are given in Appendix B.5. Similarly, the space-continuous-time-discrete weak formulations obtained from the fully discrete Störmer-Verlet weak formulations (B.22) to (B.26) are given in Appendix B.6.

Note that these weak formulations may also be obtained by taking the variations of the variational principle (3.27) with respect to h, ψ_1 and $\psi_{i'}$ (*cf.* Appendix B.7). However, the space-discrete variational principle (3.29) is required to derive the temporal evaluation of the unknowns consistently, as done in section 3.4.

Before analysing the simulations in section 3.7, an optimisation of the solvers is conducted in the next section in order to increase the computational speed of the model.

3.6 Optimisation of the solvers

3.6.1 Performance analysis

We solve our system of equations with the symplectic Euler and Störmer-Verlet schemes in the three-dimensional domain defined in Table 3.1. The wavemaker motion is set as

Domain				**Beach**	
L_x **[m]**	L_y **[m]**	H_0 **[m]**	$H(x=L_x)$ **[m]**	x_B **[m]**	s_B **[-]**
5.5	1.0	1.0	0.5	4.0	0.2
Wavemaker					
λ **[m]**	k **[rad/m]**	ω **[rad/s]**	T_w **[s]**	γ **[m]**	L_w **[m]**
2.0	3.14	5.54	1.13	0.03	1.0
Resolution					
Δx **[m]**	Δy **[m]**	N_{xy} **[-]**	N_z **[-]**	N_{tot} **[-]**	Δt **[s]**
0.05	0.05	2751	9	24759	0.004

Table 3.1: Parameters used in the test case. Dimensions are given in square brackets.

$$R(y,t) = \gamma\frac{2y - L_y}{L_y}\cos(\omega t). \tag{3.93}$$

Figure 3.6 shows the wavemaker motion (top) and velocity (red) at positions $y = 0$ and $y = L_y$. The code runs for a total of $T = 1.13$s; that is, for one period of the wavemaker. The option

```
solver_parameters = {"ksp_converged_reason":True}
```

enables to print the number of iterations for each solver and the reason for convergence. The output of this command in our case is printed for Symplectic-Euler (Fig. 3.7) and Störmer-Verlet (Fig. 3.8). For every case, convergence is reached because the tolerance threshold is satisfied, as indicated by the parameter *CONVERGED_RTOL*.

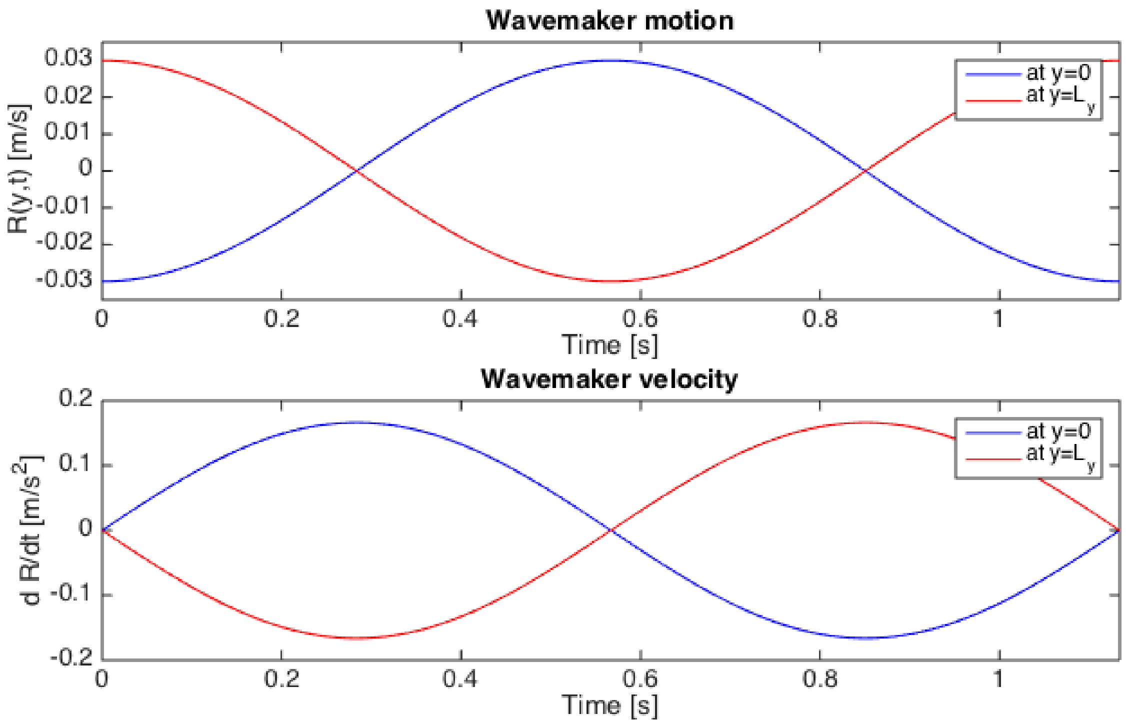
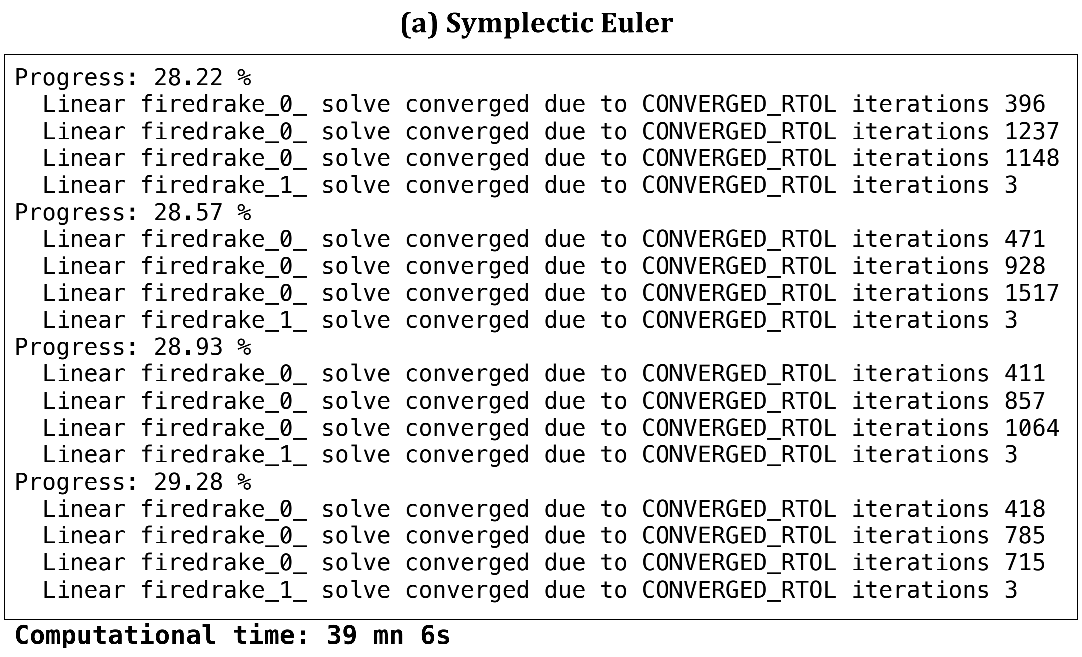


Figure 3.6: Evolution of the wavemaker motion (top) and velocity (bottom) at $y = 0$ (blue) and $y = L_y$ (red).

In the case of the Symplectic-Euler scheme (Fig. 3.7), two solvers are called at each temporal iteration: *Linear firedrake_0* and *Linear firedrake_1*. The first one, *Linear firedrake_0*, solves Eq. (B.33) and Eq. (B.34) in a mixed system in order to update h^{n+1} and $\hat{\psi}^*$ simultaneously. The process of solving these nonlinear equations is split into several linear Newton steps, each calling the linear solver *Linear firedrake_0*. This is why the *Linear firedrake_0* is called two or three times (depending on the first guess accuracy) at each temporal iteration. Figure 3.7 and Table 3.2 reveal that this linear solver requires hundreds to thousands of iterations before converging to the tolerance threshold. This iteration process consumes a considerable amount of computational time, as shown in Table 3.2. Solving the nonlinear Eqs. (B.33) and (B.34) indeed takes on average about 8s, while solving the linear Eq. (B.35) takes only 0.05s. This last equation is solved by calling the *Linear firedrake_1* solver, which converges in only 3 iterations, and hence the short computational time required.

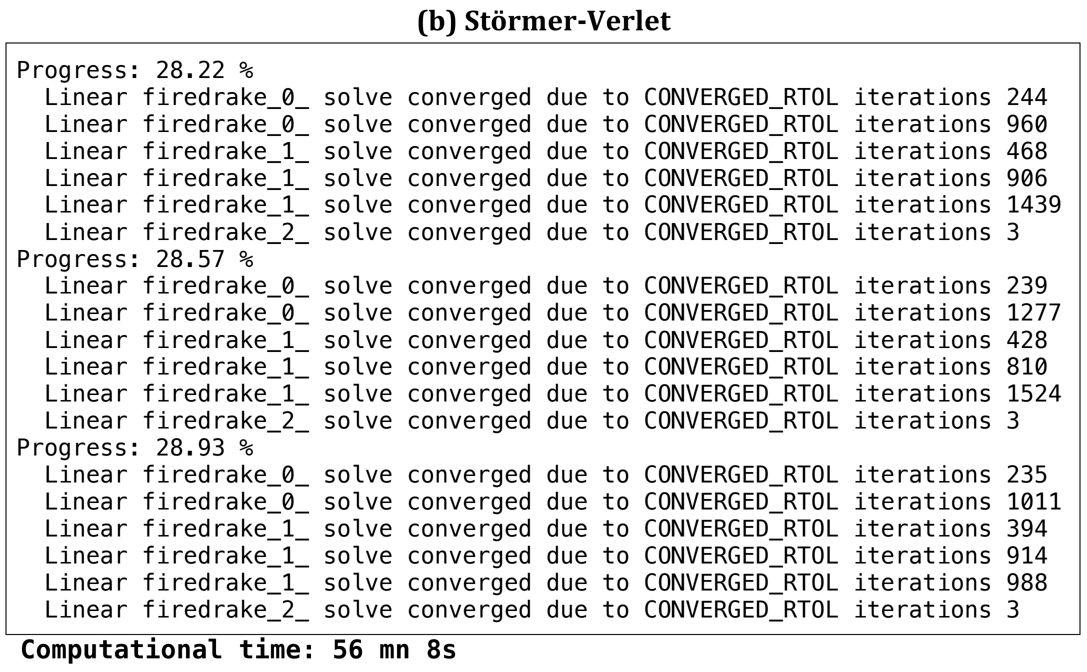

(a) Symplectic Euler

```
Progress: 28.22 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 396
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1237
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1148
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 3
Progress: 28.57 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 471
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 928
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1517
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 3
Progress: 28.93 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 411
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 857
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1064
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 3
Progress: 29.28 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 418
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 785
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 715
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 3
```

Computational time: 39 mn 6s

Figure 3.7: Number of iterations for convergence with the symplectic Euler scheme before any optimisation.

The two nonlinear and the linear steps of the Störmer-Verlet scheme are solved with three solvers, as shown in Fig. 3.8. First, *Linear firedrake_0* updates the surface velocity potential ψ_s at the intermediate time $t^{n+1/2}$, simultaneously with the sub-surface velocity potential $\hat{\psi}^{*}$, through Eqs. (B.37) and (B.38). This nonlinear step is split into two linear sub-steps, each converging after hundreds to thousands of iterations. As shown in table 3.2, this process consumes on average about 5.5s per temporal iteration of $\Delta t = 0.004s$, thus decreasing considerably the efficiency of the model. Similarly, solving Eqs. (B.39) and (B.40) to update the depth h^{n+1} and the sub-surface velocity potential $\hat{\psi}^{**}$, consists of three linear steps calling the *Linear firedrake_1* solver and converging after hundreds to thousands of iterations. Combined, these three calls converge in about 1500 iterations on average, which takes almost 7s per time step (*cf.* Table 3.2). Finally, the surface velocity potential ψ^{n+1} is computed when solving the linear Eq. (B.41) with the *Linear firedrake_2* solver. This linear weak formulation is solved in one call of the *Linear firedrake_2* solver, after 3 iterations and 0.05s on average.


(b) Störmer-Verlet

```
Progress: 28.22 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 244
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 960
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 468
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 906
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 1439
  Linear firedrake_2_ solve converged due to CONVERGED_RTOL iterations 3
Progress: 28.57 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 239
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1277
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 428
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 810
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 1524
  Linear firedrake_2_ solve converged due to CONVERGED_RTOL iterations 3
Progress: 28.93 %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 235
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 1011
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 394
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 914
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 988
  Linear firedrake_2_ solve converged due to CONVERGED_RTOL iterations 3
```

Computational time: 56 mn 8s

Figure 3.8: Number of iterations for convergence with the Störmer-Verlet scheme before any optimisation.

Scheme	**Solver**	**Av. number of iterations**	**Av. time for convergence [s]**	**% of total solving time**
Symplectic Euler	*Linear firedrake_0*	$\sim$ 2400	8.21	99.34
	Linear firedrake_1	3	5.48×10^{-2}	0.66
Störmer-Verlet	*Linear firedrake_0*	$\sim$ 1350	5.53	44.53
	Linear firedrake_1	$\sim$ 1500	6.84	55.07
	Linear firedrake_2	3	4.93×10^{-2}	0.40

Table 3.2: Averaged number of iterations and time for convergence for each solver of the symplectic Euler and Störmer-Verlet schemes before any optimisation.

From the above analysis and Table 3.2, it is now confirmed that most of the computational time is used to solve the nonlinear weak formulations. Calling the nonlinear variational solver to solve the first step of the symplectic Euler scheme, *i.e.*, Eqs. (B.33) and (B.34), indeed takes 99.22% of the total solving time, against 0.78% for the linear Eq. (B.35). Similarly, solving the first and second steps of the Störmer-Verlet scheme, respectively Eqs. (B.37)-(B.38) and Eqs. (B.39)-(B.40), uses 99.60% of the total solving time (44.53% and 55.07%, for the first and second steps respectively), against 0.40% for the linear step Eq. (B.41). More particularly, a lot of time is lost in the internal linear steps of these nonlinear variational solvers, which aim to factorise and precondition the system of equations. Decreasing the computational time thus involves setting an appropriate factorisation and preconditioning method. This solution is explained in the next section.

3.6.2 Preconditioning

Firedrake and PETSc [120, 9, 7, 71, 96] solve the linear systems with the Krylov subspace method (see, for instance, [100, 126, 61]). By default, the Firedrake solver uses the generalized minimal residual method (GMRES) based on an incomplete LU factorisation to precondition the problem. As the factorization method should be chosen depending on the system characteristics, this default solver option may be changed via the solver option *solver_parameters*. For a mixed system involving two unknowns, such as in our nonlinear weak formulations, Firedrake advises to use the PETSc's "field-split" technology, which builds preconditioners from Schur complements. As our equations are a mixed system derived from the Schur complement approach, we choose to follow this method to establish better preconditioning methods. This is done through the following command,

```
solver_parameters = {"ksp_converged_reason":True, \
                     "pc_type": "fieldsplit",\
                     "pc_fieldsplit_type": "schur",\
                     "pc_fieldsplit_schur_fact_type": "upper"}
```

where we set the preconditioner type to "fieldsplit", and the field-split's type to 'schur' (see the documentation of Firedrake [48] and PETSc [8]). The last option, *pc_fieldsplit_schur_fact_type*, sets the factorisation type for the Schur complement, which involves the computing of either the full system (option *FULL*), the first two matrices (option *lower*) or the last two matrices (option *upper*). The details on the Schur complement factorisation and relative definitions may be found in the online Firedrake documentation [48] or in the PETSc' manual [8]. After trying the three above-mentioned options, we chose the *upper* option as this was the one giving the fastest results.

For the linear weak formulations, we let Firedrake and PETSc solve the system directly by computing an LU factorisation, with the following parameters:

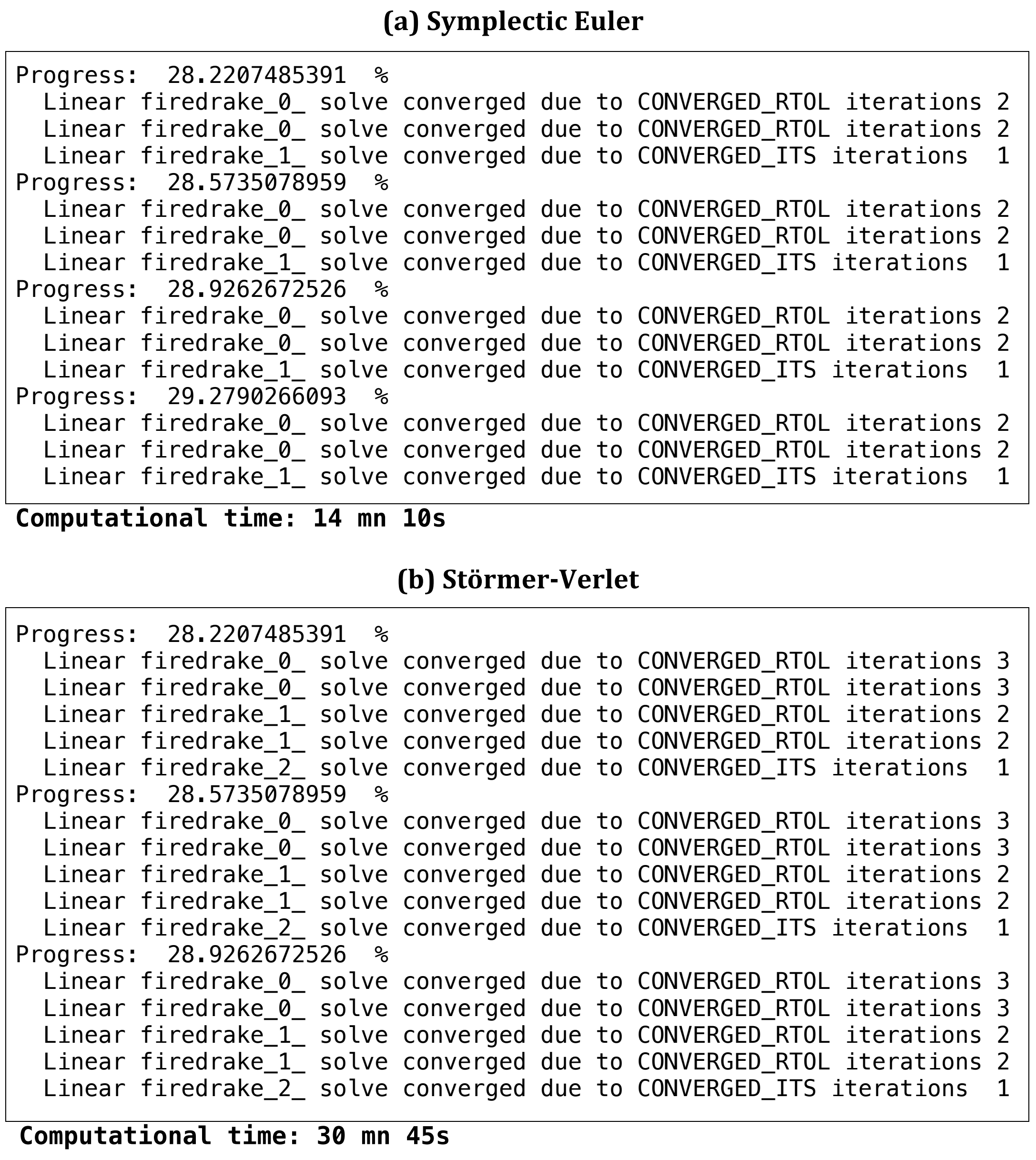
```
solver_parameters = {"ksp_converged_reason":True, 'ksp_type': 'preonly',
                           'pc_type':'lu'}
```

With an appropriate factorisation method and preconditioning, the number of iterations has been considerably reduced for each solver. Figure 3.9 indeed shows that with both (a) symplectic Euler and (b) Störmer-Verlet, the nonlinear variational solvers now require 4 to 6 iterations to converge to the tolerance threshold. This impact on the number of iterations must be analysed with caution, as it does not imply a proportional drop of the computational time. Building the preconditioning matrices indeed requires some additional time, that decreases the speed of each iteration. It is therefore essential to analyse not only the number of iterations but also the convergence time.

Scheme	**Solver**	**Av. number of iterations**	**Av. time for convergence [s]**	**% of total solving time**
Symplectic Euler	*Linear firedrake_0*	4	2.93	98.29
	Linear firedrake_1	1	5.10×10^{-2}	1.71
Störmer-Verlet	*Linear firedrake_0*	6	3.68	54.66
	Linear firedrake_1	4	3.00	44.56
	Linear firedrake_2	1	5.25×10^{-2}	0.78

Table 3.3: Averaged number of iterations and time for convergence for each solver of the symplectic Euler and Störmer-Verlet schemes with appropriate preconditioning.

Table 3.3 gives additional information about the last. Despite only 4 iterations instead of 2400 on average, the nonlinear variational solver of the symplectic Euler scheme still consumes 98.29% of the solving time, with on average a bit less than 3s to converge. This convergence time is 2.8 times



(a) Symplectic Euler

```
Progress:  28.2207485391  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_ITS iterations  1
Progress:  28.5735078959  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_ITS iterations  1
Progress:  28.9262672526  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_ITS iterations  1
Progress:  29.2790266093  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_ITS iterations  1
```

Computational time: 14 mn 10s

(b) Störmer-Verlet

```
Progress:  28.2207485391  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_2_ solve converged due to CONVERGED_ITS iterations  1
Progress:  28.5735078959  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_2_ solve converged due to CONVERGED_ITS iterations  1
Progress:  28.9262672526  %
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_0_ solve converged due to CONVERGED_RTOL iterations 3
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_1_ solve converged due to CONVERGED_RTOL iterations 2
  Linear firedrake_2_ solve converged due to CONVERGED_ITS iterations  1
```

Computational time: 30 mn 45s

Figure 3.9: Number of iterations for convergence for (a) Symplectic Euler and (b) Störmer-Verlet with appropriate preconditioning.

faster than without preconditioning, which considerably speeds the model up. Similarly, the nonlinear solvers of the Störmer-Verlet scheme are now 1.5 and 2.3 times faster than without preconditioning, which consequently results in a total computational time of 30 minutes instead of 56 minutes, thus 1.9 times faster. The linear solver used to update ψ_1^{n+1}, both with symplectic Euler and Störmer-Verlet, has also slightly been improved by the LU factorisation.

For a large number of elements, our Firedrake code *3D_Tank.py* may also be run in parallel on several cores through a MPI call :

```
mpirun -n 16 3D_Tank.py
```

One advantage of Firedrake is that the code does not need to be changed for running in parallel: the internal functions are built to support parallel runs. This feature considerably speeds up the process. Three-dimensional simulations are shown and analysed in the next section.

3.7 Convergence analysis

Now that the solvers are optimised, their temporal and spatial accuracy may be checked. In this section, we verifiy the temporal convergence of the solvers through the energy conservation, and the spatial convergence through a test of convergence.

3.7.1 Energy conservation

In section 3.4, we derived the temporal schemes in a way that ensures stability and overall energy conservation and thus in principle eliminates numerical dissipation of energy. Based on the parameters given in Table 3.1, we compare the energy fluctuations for two time steps, $\Delta t_1 = 0.001s$ and $\Delta t_2 = 0.002s$, using each temporal scheme, Symplectic-Euler and Störmer-Verlet.

	Symplectic Euler	**Störmer-Verlet**
$\Delta t_1 = 0.001s$	Case SE1	Case SV1
$\Delta t_2 = 0.002s$	Case SE2	Case SV2

Table 3.4: Tests for which the energy variations are computed.

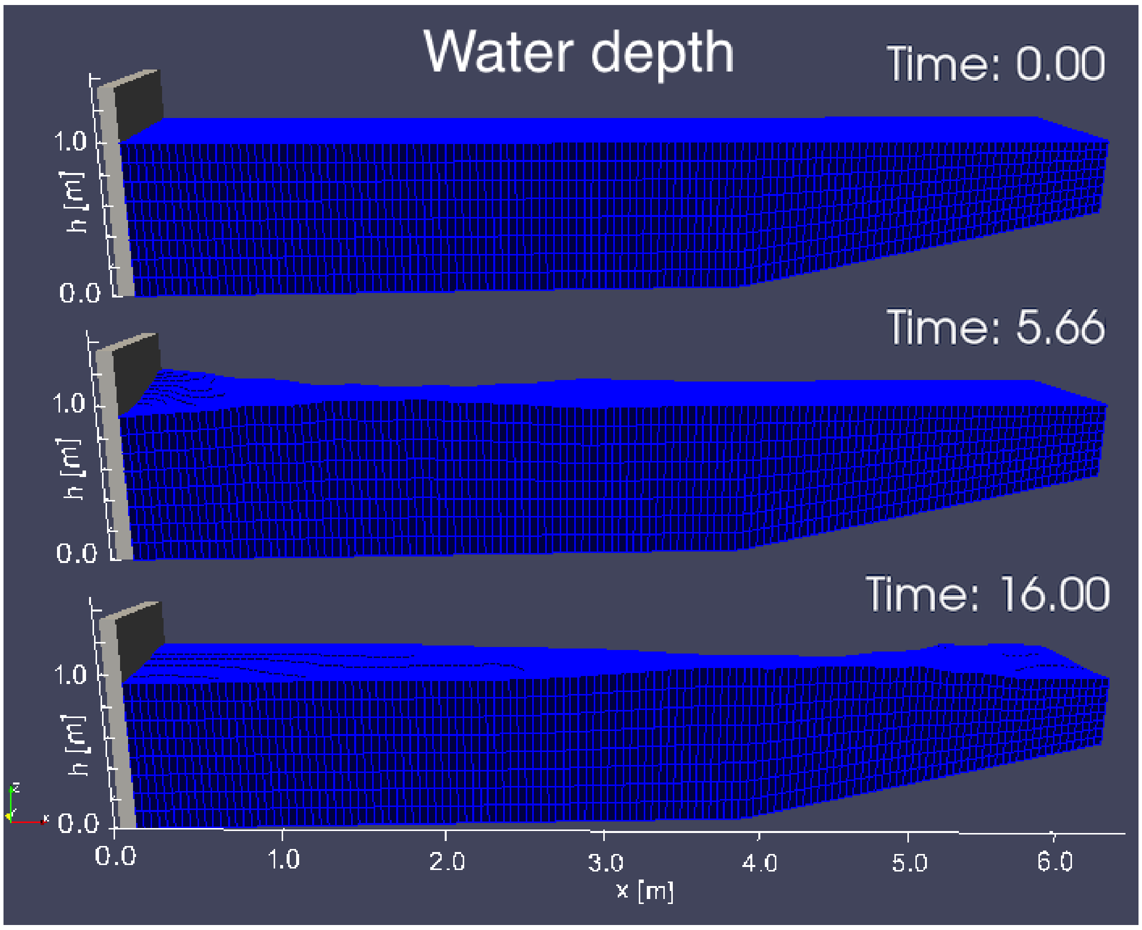


Figure 3.10: Snapshots of the free-surface elevation at $t = 0.0$s (top), $t = 5.66$s (middle) and $t = 16$s (bottom), obtained with the symplectic-Euler scheme and $\Delta t = 0.001$s.

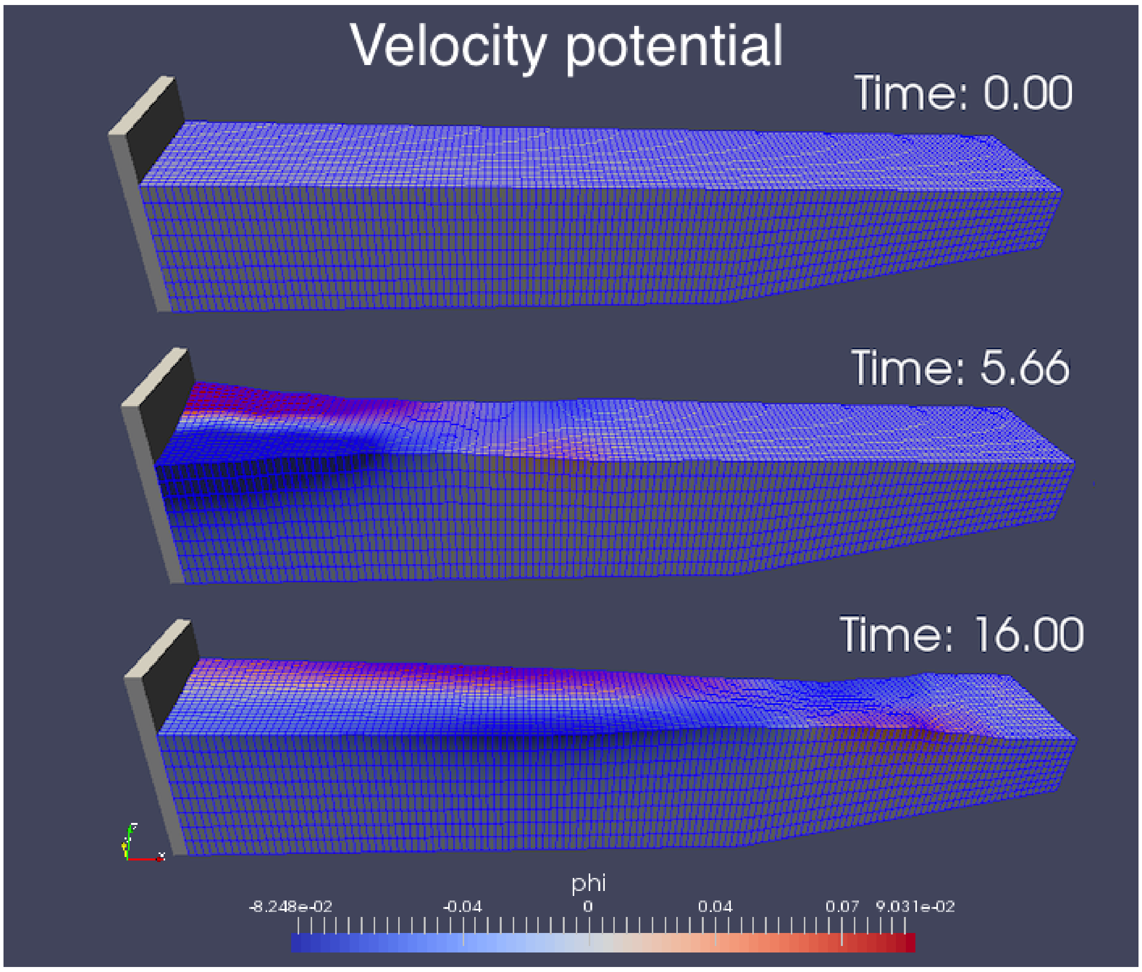


Figure 3.11: Snapshots of the velocity potential at $t = 0.0$s (top), $t = 5.66$s (middle) and $t = 16$s (bottom), obtained with the symplectic-Euler scheme and $\Delta t = 0.001$s.

The wavemaker motion, defined by Eq. (3.93), is shown in Fig. B.4 at various locations in y, together with its velocity (see in the Appendix B.8). Figures 3.10 and 3.11 show snapshots of the water depth and velocity potential respectively, at various times. At time $t = 0$ (top), the water is at rest and the wavemaker is off. We then turn it on for five wave periods, *i.e.*, for 5.65s, which results in free-surface motion in the basin (middle). At $t = 5.65$s, the wavemaker is turned off, leading to calm water again (bottom). We stop the simulations after a total of 17s. For optimised computational time, the code is run in parallel on 16 cores.

Figure 3.12 shows the energy evolution with (a) symplectic Euler (cases SE1 and SE2) and (b) Störmer-Verlet (cases SV1 and SV2). At $t = 0.0s$ the wavemaker is off and the water is at rest (*cf.* top line in Fig. 3.10), hence there is no kinetic energy in the system and the potential energy is offset to be zero too. When the wavemaker motion starts, energy is given to the system and energy thus increases until $t = 5.65$s, when the wavemaker is turned off again. This net gain of energy occurs because the wavemaker leads to a net energy input into the system. Indeed, due to the wavemaker, the Hamiltonian (3.32) depends explicitly on time. As a consequence,

$$\frac{dH(\boldsymbol{p_1},\boldsymbol{h},\boldsymbol{\psi_{i'}},t)}{dt} = \frac{\partial H}{\partial \boldsymbol{p_1}}\frac{\partial \boldsymbol{p_1}}{\partial t} + \frac{\partial H}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial t} + \frac{\partial H}{\partial \boldsymbol{\psi_{i'}}}\frac{\partial \boldsymbol{\psi_{i'}}}{\partial t} + \frac{\partial H}{\partial t}. \tag{3.94}$$

Substituting equations (3.33) into (3.94), we obtain that

$$\frac{dH(\boldsymbol{p_1},\boldsymbol{h},\boldsymbol{\psi_{i'}},t)}{dt} = \frac{\partial H}{\partial t} \neq 0, \tag{3.95}$$

and therefore the energy is not conserved. However, when the wavemaker is turned off, then

$$\frac{dH(\boldsymbol{p_1},\boldsymbol{h},\boldsymbol{\psi_{i'}},t)}{dt} = \frac{dH(\boldsymbol{p_1},\boldsymbol{h},\boldsymbol{\psi_{i'}})}{dt} = \frac{\partial H}{\partial \boldsymbol{p_1}}\frac{\partial \boldsymbol{p_1}}{\partial t} + \frac{\partial H}{\partial \boldsymbol{h}}\frac{\partial \boldsymbol{h}}{\partial t} + \frac{\partial H}{\partial \boldsymbol{\psi_{i'}}}\frac{\partial \boldsymbol{\psi_{i'}}}{\partial t} = 0, \tag{3.96}$$

and the overall energy is conserved. This is what is observed in Fig. 3.12, for $t > 5.65s$, and in Fig. 3.10 (bottom) where waves have not been dampened despite more than 10s ($\approx$ nine wave periods) without any wavemaker motion. Therefore, the energy conservation in the absence of wavemaker is verified, as no drift is observed after switching off the wavemaker.

In Fig. 3.13, we verify the consistency of the temporal schemes by focusing on the energy variations after the wavemaker motion has been switched off (*i.e.*, for $5.65s \leq t \leq 17s$). The Hamiltonian dynamics of our temporal schemes result in bounded and small amplitude energy oscillations, which confirms that the overall energy is conserved with both (a) symplectic Euler and (b) Störmer-Verlet. The amplitude of these oscillations depends on the time step.

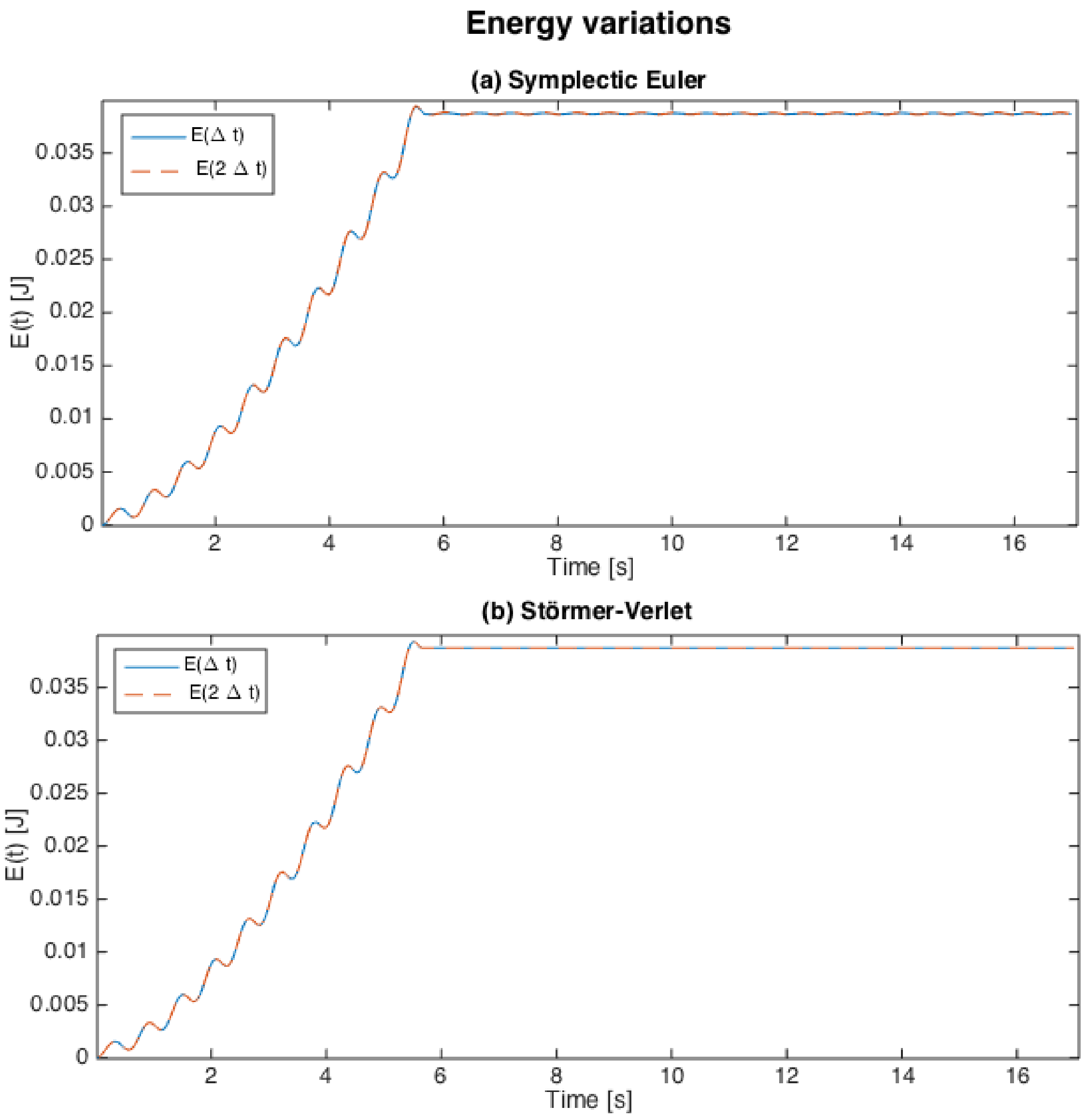


Figure 3.12: Energy variations with (a) the 1^{st}-order symplectic Euler scheme and (b) the 2^{nd}-order Störmer-Verlet scheme. The wavemaker generates the waves from $t = 0.0$s to $t = 5.65s$ and is then turned off. The simulations are computed with $\Delta t_1 = 0.001s$ (full line) and $\Delta t_2 = 2\Delta t_1 = 0.002s$ (dashed line).

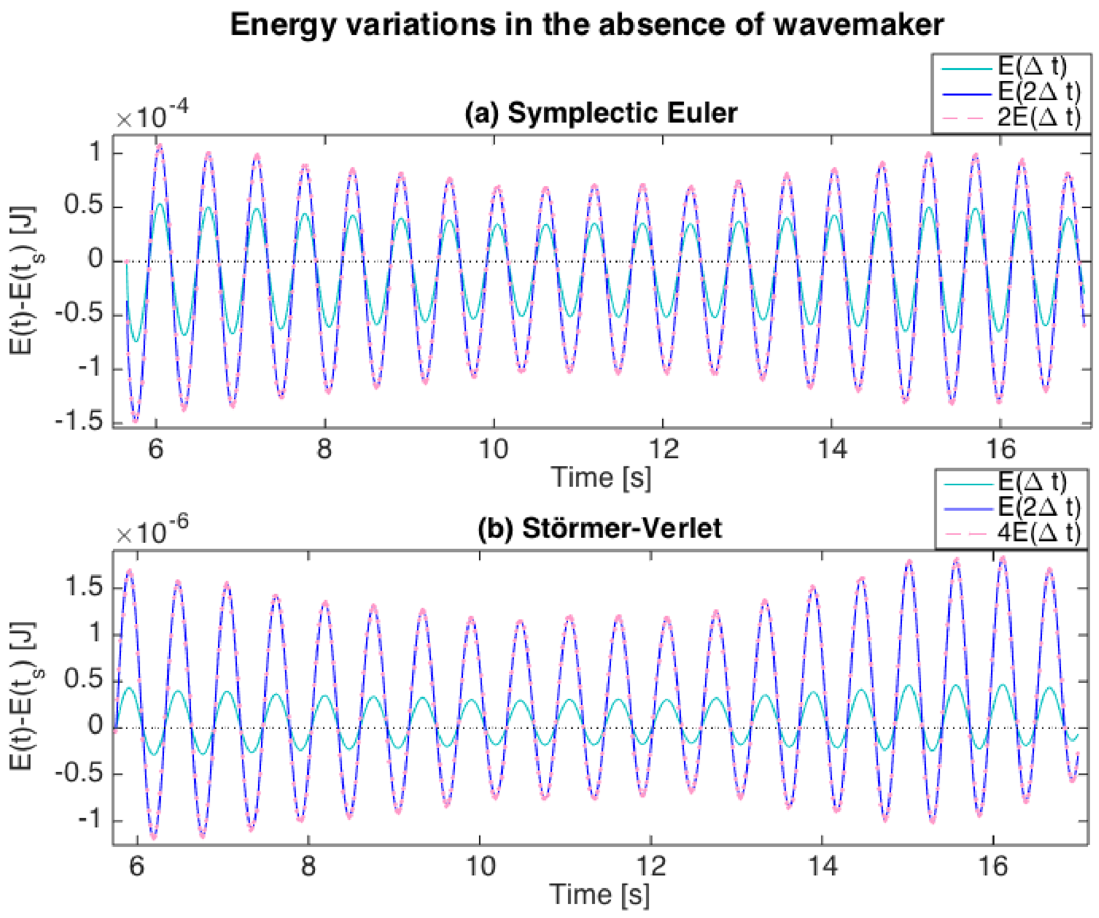


Figure 3.13: Energy variations with (a) the 1^{st}-order symplectic Euler scheme and (b) the 2^{nd}-order Störmer-Verlet scheme in the absence of wavemaker motion. The full dark green lines show variations in the cases SE1 (a) and SV1 (b), the full blue lines show variations in the cases SE2 (a) and SV2 (b), and the dashed pink lines are respectively twice the variations of SE1 (a) and four times the variations of SV1 (b).

With the symplectic Euler scheme, the variations ΔE_{SE} in the case SE2 are twice larger than in the case SE1, that is

$$\Delta E_{SE}(\Delta t_2) = 2\Delta E_{SE}(\Delta t_1), \quad \text{for } \Delta t_2 = 2\Delta t_1. \tag{3.97}$$

This is a consequence of the order of the discretisation, which is 1^{st}-order in the symplectic-Euler scheme, and thus results in a linear increase of the energy oscillations with the time step.

However, for the 2^{nd}-order Störmer-Verlet scheme we observe that in the case SV2 the energy variations ΔE_{SV} are four times larger than those in the case SV1:

$$\Delta E_{SV}(\Delta t_2) = 4\Delta E_{SV}(\Delta t_1), \quad \text{for } \Delta t_2 = 2\Delta t_1. \tag{3.98}$$

This quadratic increase of the energy oscillations when doubling the time step is consistent with the fact that Störmer-Verlet is a 2^{nd}-order scheme.

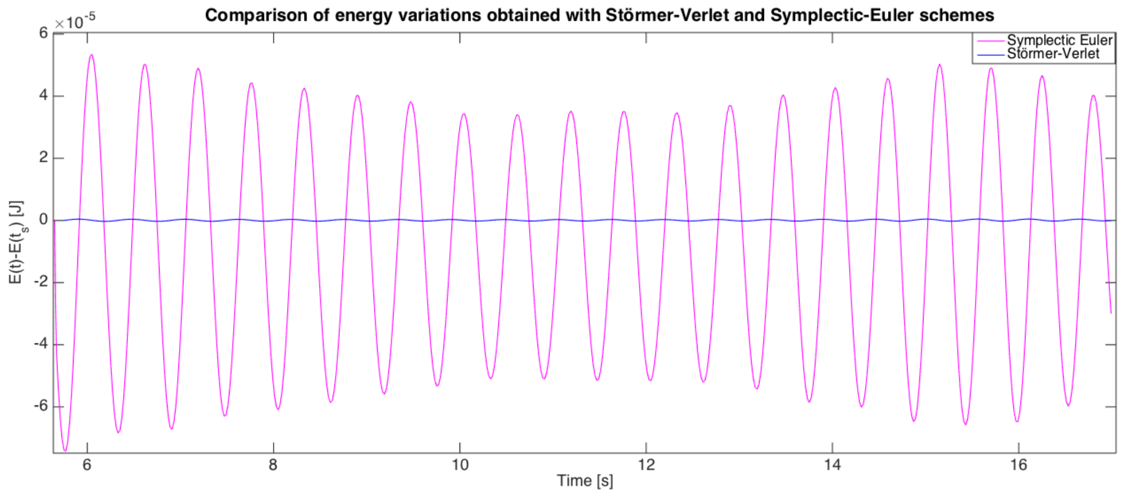


Figure 3.14: Comparison of energy variations with the 1^{st}-order symplectic-Euler scheme (pink) and the 2^{nd}-order Störmer-Verlet scheme (blue) with $\Delta t = 0.001$s to highlight the higher accuracy of the Störmer-Verler scheme.

In terms of computational time, the symplectic-Euler scheme is slighly more than twice faster than Störmer-Verlet. On 16 cores, cases SE1 and SV1 were run in 1h47min and 3h44min respectively (*i.e.*, a 2.1 ratio), and cases SE2 and SV2 in 54min and 2h10min respectively (*i.e.*, a 2.4 ratio). Similarly, a ratio of 2.17 was obtained when running on one core in section 3.6.2 (*cf.* Fig. 3.9). However, the accuracy of the Störmer-Verlet scheme is much higher than the one of the symplectic-Euler scheme, as the error decreases quadratically with the time step. Figures 3.13 and 3.14 indeed show that for the same time step, the energy variations obtained with Störmer-Verlet are about 100 times smaller than those obtained with symplectic-Euler (order 10^{-6} *vs* order 10^{-4} respectively). As symplectic-Euler is a 1st–order scheme, the time step would need to be 100 times smaller to get oscillations of order 10^{-6} and thus the same accuracy as with Störmer-Verlet. Consequently, the computational time would be about 100 times longer, so the same accuracy is reached about 50 times faster with Störmer-Verlet than with symplectic Euler. One therefore needs to carefully choose the time scheme depending on requirements. If the objective is to get fast simulations with good-but-not-excellent accuracy, then the symplectic-Euler scheme is a better option. However, if the objective is to minimise the error, then Störmer-Verlet is definitely the best choice.

3.7.2 Test of spatial convergence

In space, the approximation with 1st–order continuous Galerkin polynomials should be of second-order accuracy [130]. To verify the spatial accuracy, we solve the nonlinear potential-flow equations in a domain with dimensions given in Table 3.5, time step $\Delta t = 0.001$s, 9 elements in the vertical, and various horizontal spatial resolutions, as shown in Fig. 3.15.

Domain				**Beach**	
L_x **[m]**	L_y **[m]**	H_0 **[m]**	$H(x = L_x)$ **[m]**	x_B **[m]**	s_B **[-]**
4.0	0.8	1.0	0.5	1.5	0.2
Wavemaker					
λ **[m]**	k **[rad/m]**	ω **[rad/s]**	T_w **[s]**	γ **[m]**	L_w **[m]**
2.0	3.14	5.54	1.13	0.02	0.8

Table 3.5: Parameters used in the test case. Dimensions are given in square brackets.

First, the spatial resolution is set to $\Delta y = \Delta x = 0.025$m and the obtained estimation of the water depth is used as reference value h_{ex}. The error between this reference value and the depth $h_{0.05}$ and $h_{0.1}$, computed with resolutions $\Delta y = \Delta x = 0.05$m and $\Delta y = \Delta x = 0.1$m respectively, is then computed as the L^2– and L^∞–norms of their difference, that is

$$\text{err}_{L^2}(h^n) = ||h^n(x,y) - h^n_{ex}(x,y)||_2 = \sqrt{\sum_i (h^n_i - h^n_{ex_i})^2}, \tag{3.99}$$

$$\text{err}_{L^\infty}(h^n) = ||h^n(x,y) - h^n_{ex}(x,y)||_\infty = \max_i |h^n_i - h^n_{ex_i}|, \tag{3.100}$$

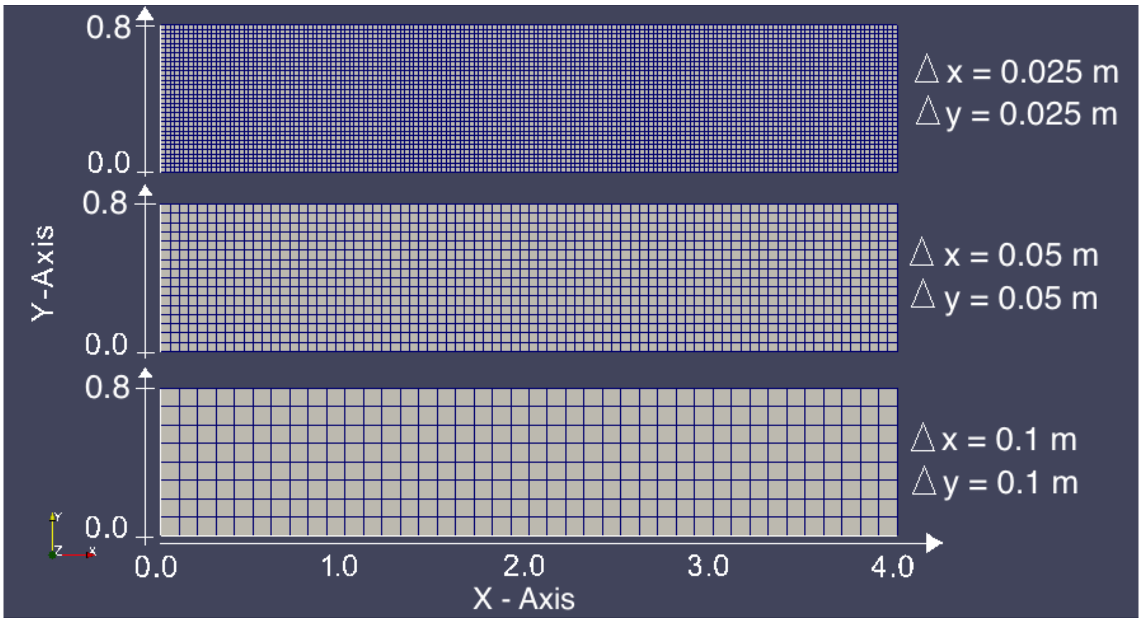


Figure 3.15: Top view of the numerical domain at time $t = 0.0$s with resolutions $\Delta x = \Delta y = 0.025$m (top),$\Delta x = \Delta y = 0.05$m (middle) and $\Delta x = \Delta y = 0.1$m (bottom).

where i designates the common nodes between the three meshes, that is, the nodes of the largest mesh. For a n^{th}–order accuracy in space, we should get

$$\text{err}(h_{2\Delta x}) = 2^n \text{err}(h_{\Delta x}). \tag{3.101}$$

As a consequence,

$$\begin{aligned}
&\frac{\text{err}(h_{2\Delta x})}{\text{err}(h_{\Delta x})} = 2^n \\
&\Rightarrow \log \frac{\text{err}(h_{2\Delta x})}{\text{err}(h_{\Delta x})} = \log(2^n) \\
&\Rightarrow \log\left(\text{err}(h_{2\Delta x})\right) - \log\left(\text{err}(h_{\Delta x})\right) = n \log(2) \\
&\Rightarrow \frac{\log\left(\text{err}(h_{2\Delta x})\right) - \log\left(\text{err}(h_{\Delta x})\right)}{\log(2)} = n.
\end{aligned} \tag{3.102}$$

Moreover,

$$\log(2) = \log(\frac{2\Delta x}{\Delta x}) = \log(2\Delta x) - \log(\Delta x), \tag{3.103}$$

so the order n of spatial convergence may be obtained via the ratio

$$n = \frac{\log\left(\text{err}(h_{2\Delta x})\right) - \log\left(\text{err}(h_{\Delta x})\right)}{\log(2\Delta x) - \log(\Delta x)}, \tag{3.104}$$

which, by definition, is the slope of the curve of $(\log(err(h_{\Delta x})), \log(err(h_{2\Delta x})))$ against $(\log(\Delta x), \log(2\Delta x))$. We therefore compute the coefficients

$$\beta^n_{L^2} = \frac{\log(err_{L^2}(h^n_{0.1})) - \log(err_{L^2}(h^n_{0.05}))}{\log(0.1) - \log(0.05)}, \tag{3.105}$$

$$\text{and } \beta^n_{L^\infty} = \frac{\log(err_{L^\infty}(h^n_{0.1})) - \log(err_{L^\infty}(h^n_{0.05}))}{\log(0.1) - \log(0.05)}, \tag{3.106}$$

for $t \in [0, 7]$s, and check that they both converge towards the expected order of convergence $n = 2$. The left column of Fig. 3.16 shows the temporal evolution of β_{L^∞} (top) and β_{L^2} (bottom) and and their convergence towards $\beta \approx 2$. This 2^{nd}-order spatial convergence is confirmed by the right column of Fig. 3.16 on which the log of the L^∞ (top) and L^2 (bottom) errors is plotted against the log of their respective spatial resolution, together with the curves of slope $\bar{\beta}_{L^\infty}$ and $\bar{\beta}_{L^2}$, computed by averaging β_{L^∞} and β_{L^2} between $t = 4.0$s and $t = 7.0$s.

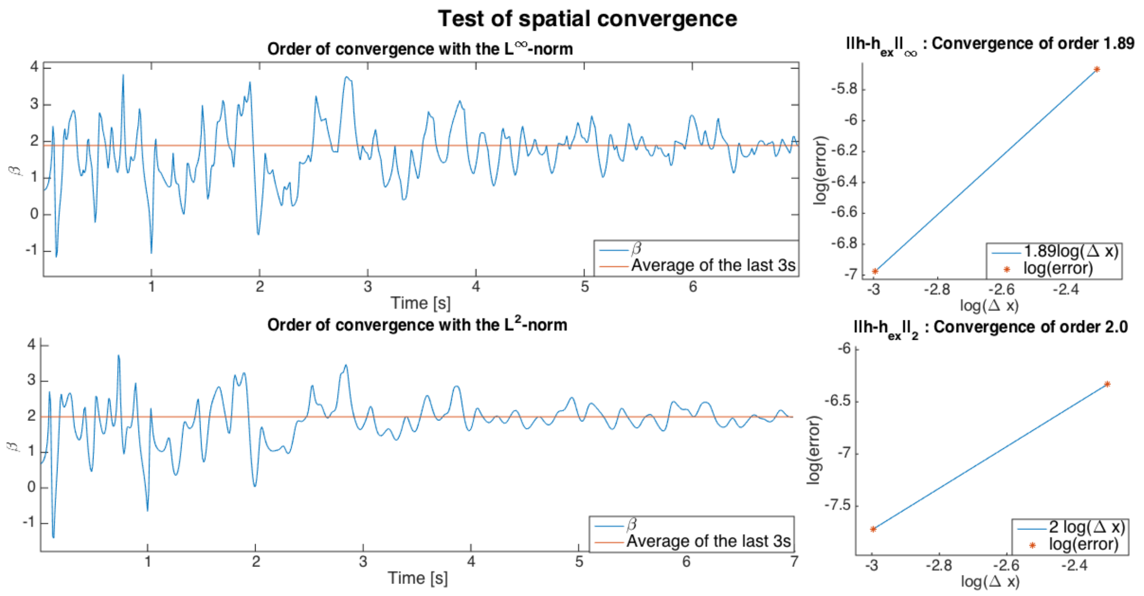


Figure 3.16: Temporal evolution of the slope of the regression line for symplectic Euler and Störmer-Verlet.

As expected the coefficients β_{L^2} and β_{L^∞} converge to $n \approx 2.0$, confirming the convergence rate of 2.0. The accuracy of the solvers is thus verified in both space and time. In the next section, we test their performance in the simulation of extreme waves.

3.8 Rogue-type wave simulations

Experiments were conducted in the shallow-water basin of MARIN (*cf* test case 202002 in [24]), which includes piston wavemakers and a flat bottom, with a depth at rest $H(x) = H_0 = 1.0$m (*cf.* Fig. 3.17). Several wave groups of various steepness were generated in order to generate a focussed wave (*cf.* section 3.8.2). Probes were placed at various locations $x_1 = 10m$, $x_2 = 20m$, $x_3 = 40m$, $x_4 = 49.5m$, $x_5 = 50m$ and $x_6 = 54m$ from the wavemaker in order to measure the free-surface elevation. These data as well as the wavemaker motion and velocity were recorded at a frequency of $50Hz$. As in [51], these measurements are used to initialise and validate our numerical model in the vertical plane (2D).

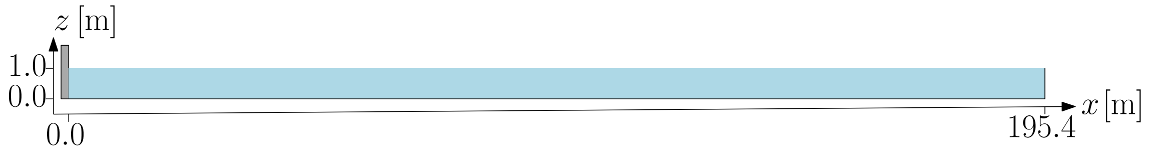


Figure 3.17: Schematic of MARIN's basin used for measurements. The tank is 195.4m long, with a constant water depth at rest of 1.0m. A piston-type wavemaker moves around $x = 0$m to generate the waves.

3.8.1 Import data from measurements

In order to generate the same wave spectra as those in the shallow-water basin of MARIN, the measured wavemaker motion and velocity are interpolated to be assigned to the corresponding numerical functions at each time step. To meet the CFL condition (3.82), the time step Δt used in our simulations must be smaller than the one used to record the data, $\Delta t\_data = 1/50s$. To use the measured wavemaker motion and velocity at each time step, we interpolate them with 1^{st}–order polynomials in each measured time interval $[t_1, t_2]$. Therefore, at time t, the interpolated motion $R_{int}(t)$ and velocity $u_{int}(t)$ of the measured motion R_{dat} and measured velocity u_{dat} are obtained through

$$R_{int}(t) = \frac{(t-t_1)R_{dat}(t_2) - (t-t_2)R_{dat}(t_1)}{t_2 - t_1}, \tag{3.107a}$$

$$u_{int}(t) = \frac{(t-t_1)u_{dat}(t_2) - (t-t_2)u_{dat}(t_1)}{t_2 - t_1}. \tag{3.107b}$$

The interpolations (3.107) are updated with time and numerically assigned to the wavemaker motion and velocity functions. Note that as we consider 2D vertical waves, the y–derivative of the wavemaker is null, that is $\partial_y R = 0$. Figure 3.18 shows that the interpolated motion and velocity of the wavemaker indeed fits the initial measurements.

The numerical free-surface elevation resulting from this wavemaker input signal may then be saved and compared to the experimental data. The results are analysed in the next section.

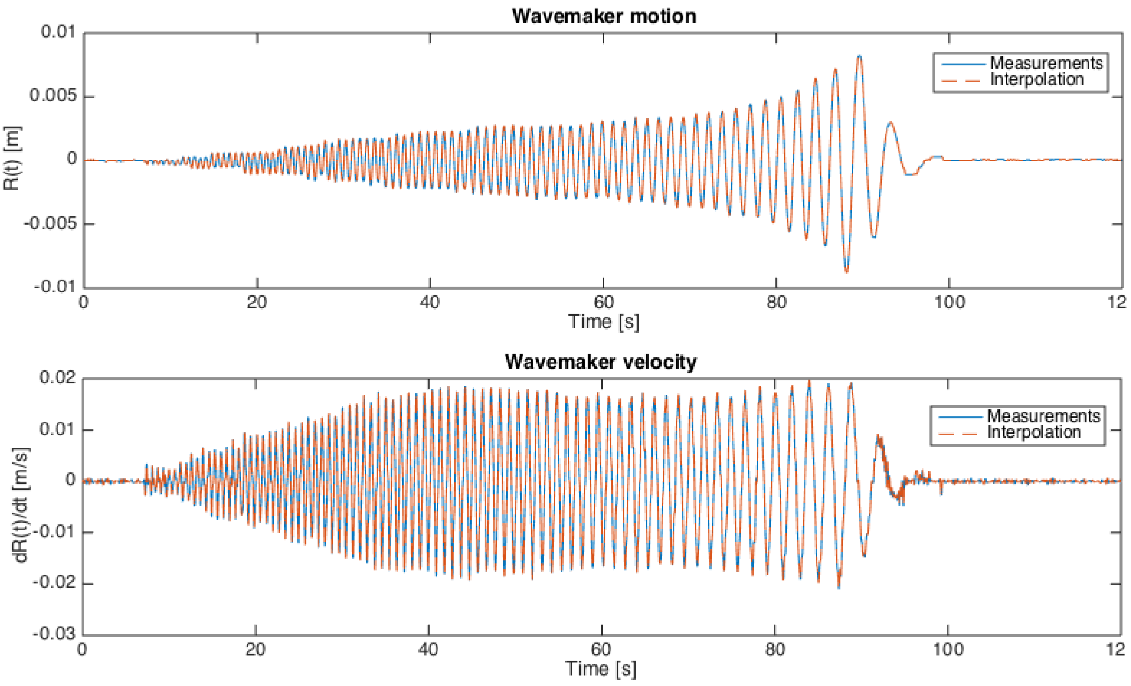


Figure 3.18: Interpolated and measured wavemaker motion (top) and velocity (bottom) in the case of focussed wave generation.

3.8.2 Focussed wave: the dispersion effect

The wavemaker input is now used to simulate a focussed wave. The wavemaker starts at rest and oscillates with various amplitudes and frequencies (*cf.* Fig. 3.18). Due to dispersion, the latest generated waves, which are longer, travel faster than the first waves, so that they all interact at a specific position to result in a focussed wave. In order to capture this focussed wave, probes are placed around the target area, at $x = 49.5$m and $x = 50$m.

Numerically, we define a 100m long basin, with flat seabed ($b(x) = 0$) and water depth at rest $H(x) = H_0 = 1.0$m. The fast Fourier transform of the measured wave signals in Fig. B.5 (see Appendix B.9) shows that the maximal relevant frequency is about $\omega \approx 18$Hz. The shortest wavelength may thus be estimated from the linear dispersion relation

$$\omega = \sqrt{\frac{2\pi}{\lambda}\tanh(\frac{2\pi}{\lambda}H_0)}, \tag{3.108}$$

which leads to $\lambda \approx 0.19$m. As explained in section 3.4.4, stability requires $\Delta x \leq \lambda$, so that the full length of the wave can be captured by the mesh. However, to ensure accurate results the wavelength should be evaluated over more than one node. To increase accuracy of the model, the mesh resolution is set to $\Delta x = \min \lambda/20 = 0.01$m. As a consequence, the time step must satisfy

$$\Delta t \leq \frac{2}{\sqrt{g\frac{2\pi}{\Delta x}\tanh(\frac{2\pi}{\Delta x}H_0)}} = 0.025\text{s}. \tag{3.109}$$

To increase accuracy, we set $\Delta t = \max \Delta t/20 = 0.001$s, which is also the time step used in [51]. Finally, the depth is split into 9 horizontal layers ($\Delta z = 0.125$m), meaning that the system counts nine unknowns for the velocity (ψ_1 at the surface and $\psi_{2,..9}$ in depth), additionally to the depth h. While the simulations are in the 2D vertical plane, the numerical results are expanded in the y–direction to be observed in 3D. The measured wave elevation is compared with the one obtained in our numerical simulations.

Figures 3.19 and 3.20 show snapshots of the free-surface elevation and velocity potential respectively, obtained with the symplectic-Euler scheme. First, from $t_0 = 0.0$s to $t_1 = 93.02$s, waves with increasing length and amplitude are generated. At time $t_2 = 105.12$s, it is evident that the waves are closer to each other than when initially generated: the dispersion effect makes the longest waves travel faster than the shortest ones. At time $t_3 = 109.40$s, the last waves have caught the first ones, leading to a unique, much higher and steeper wave: a focussed wave, which plays the role of a freak wave. Immediately after time t_3, the longest, fastest waves overtake the shortest ones (*e.g.* at times $t_4 = 113.68$s and $t_5 = 119.98$s), and the waves split again, leading to a mirror-configuration relative to the focussed wave (*e.g.* compare the snapshot at time t_2 with the one at time t_4).

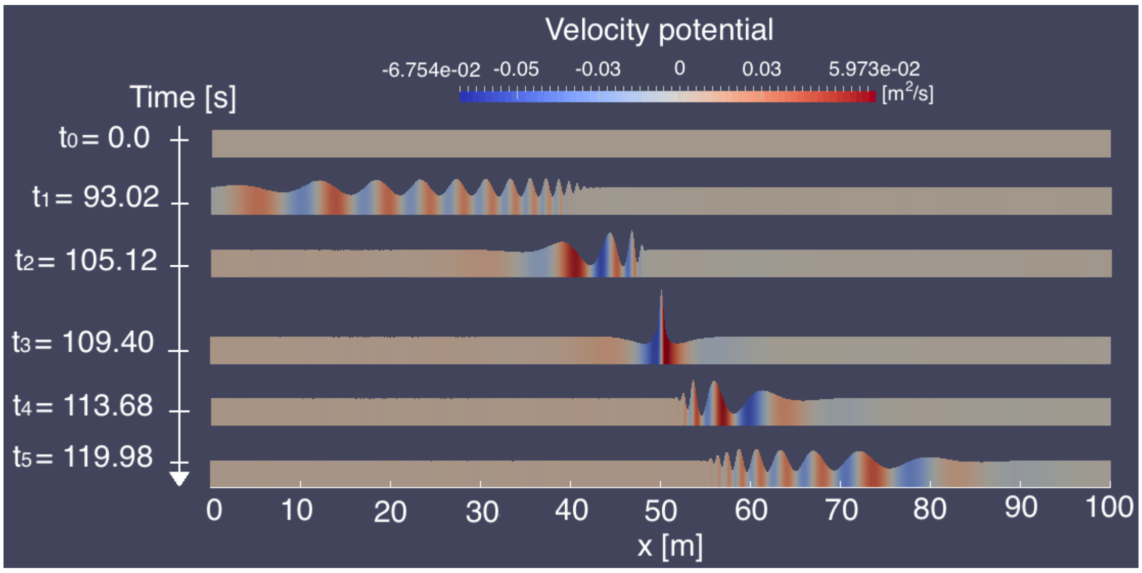


Figure 3.19: Snapshots of the velocity potential at times $t_0 = 0.0$s, $t_1 = 93.01$s, $t_2 = 105.12$s, $t_3 = 109.40$s, $t_4 = 113.68$s and $t_5 = 119.98$s. The focussed wave is captured at time $t_3 = 109.40$s.

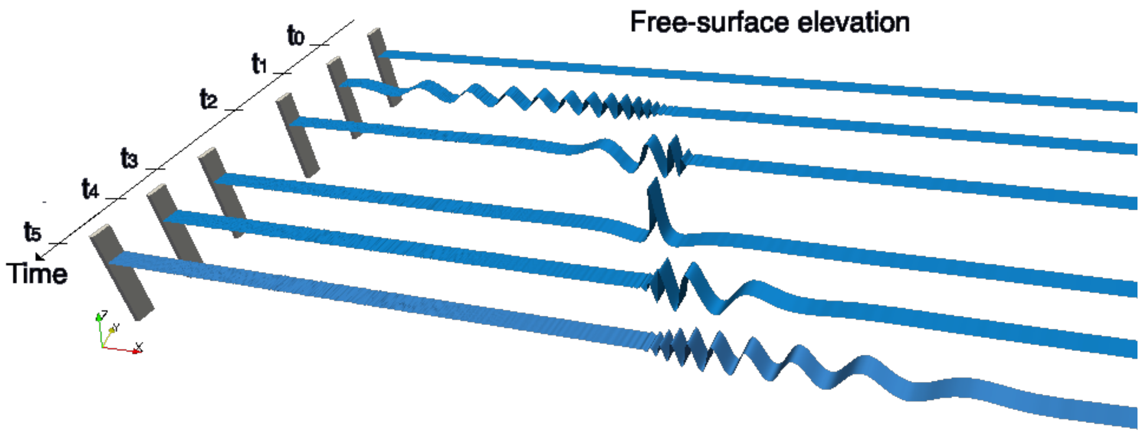


Figure 3.20: Snapshots of the free-surface elevation (bottom), at times $t_0 = 0.0$s, $t_1 = 93.01$s, $t_2 = 105.12$s, $t_3 = 109.40$s, $t_4 = 113.68$s and $t_5 = 119.98$s. The focussed wave is captured at time $t_3 = 109.40$s.

Figure 3.21 compares the measured (red) time evolution of the wave elevation at the probes with the numerical evolution obtained with the 1^{st}–order symplectic Euler scheme (dark blue) and the 2^{nd}–order Störmer-Verlet scheme (cyan). Both schemes yield agreement with measurements with an error of $O(\text{mm})$, while the wave height is of $O(\text{cm})$. The freak-wave phase and location agree with the measurement, meaning that the model may be used to simulate waves in a target area up to at least 50m from the wavemaker. Figure 3.22 shows that all the experimental frequency modes are well captured by the numerical model.

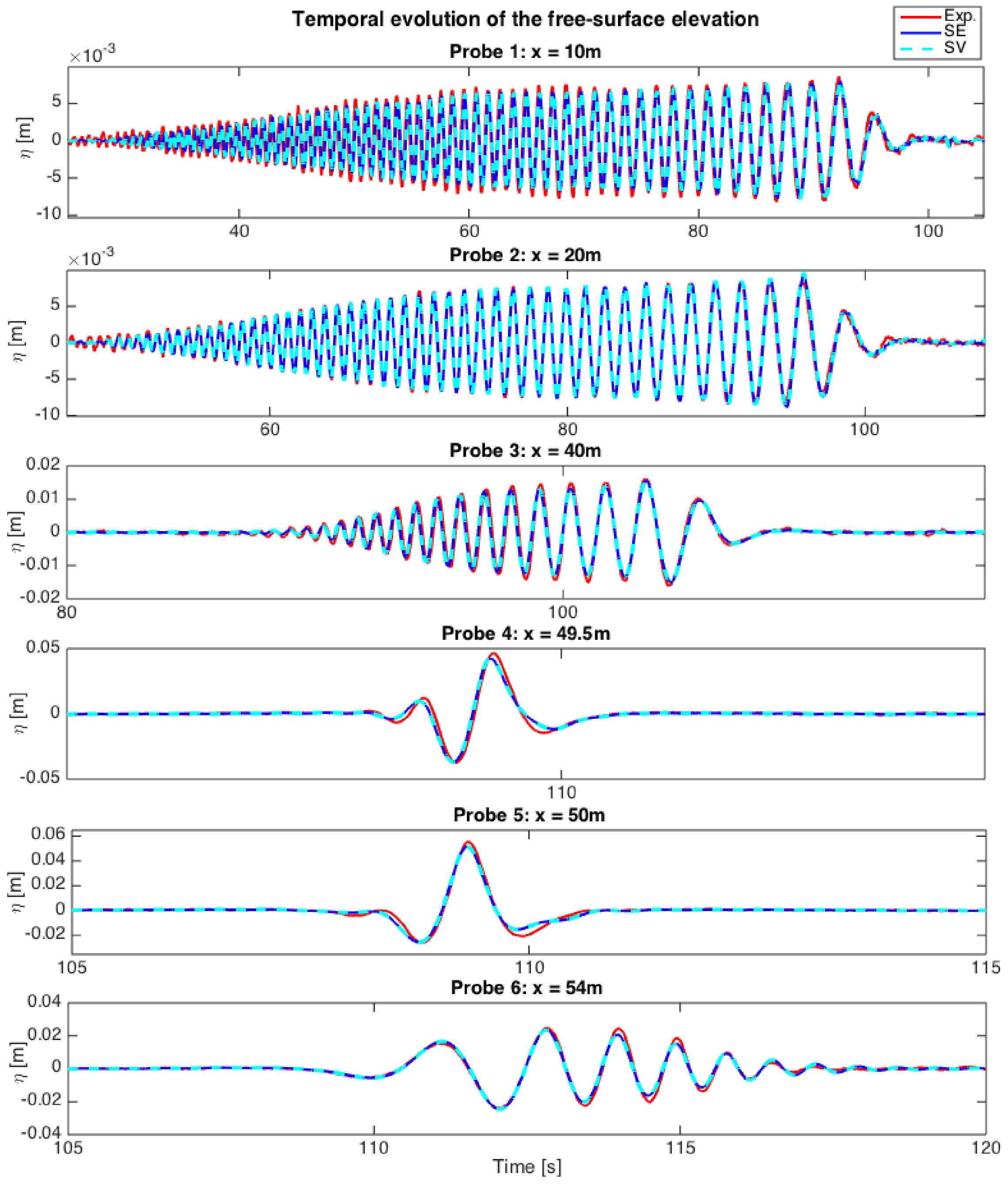


Figure 3.21: Wave elevations of numerical (blue and cyan) and experimental (red) data at various locations. The numerical results are obtained with the symplectic Euler scheme (blue full line) and the Störmer-Verlet scheme (cyan dashed-dotted line).

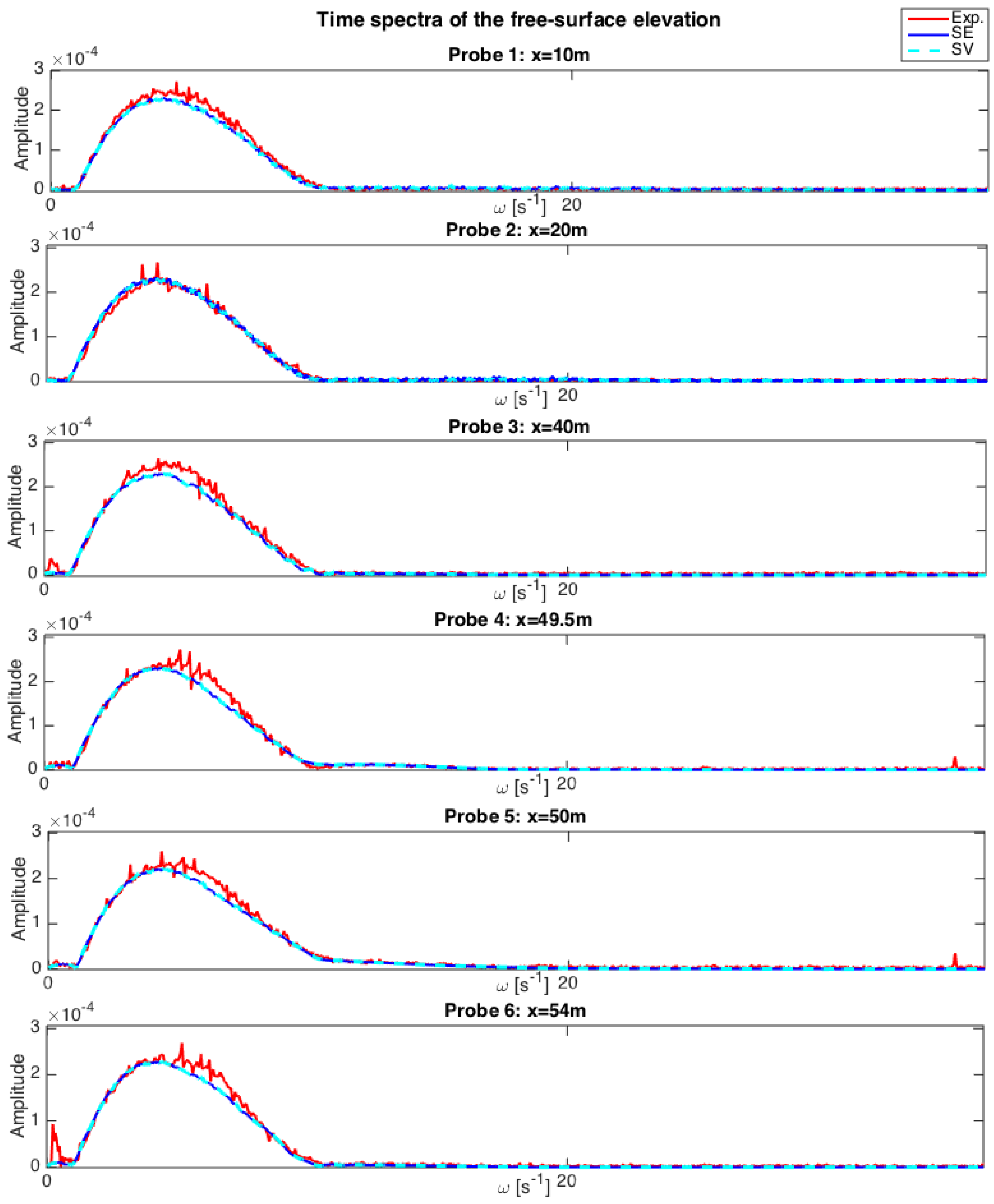


Figure 3.22: Fast Fourier transform of the wave elevations of numerical (blue and cyan) and experimental (red) data at various locations. The numerical results are obtained with the symplectic Euler scheme (blue full line) and the Störmer-Verlet scheme (cyan dashed line).

3.9 Conclusions and extensions

3.9.1 Advantages of the present numerical wave tank

The methodology presented in this chapter has led to a three-dimensional nonlinear potential-flow model that has overcome four main industrial and computational challenges.

First, we have checked the model ***accuracy*** through validation against existing experiments at MARIN. The simulations not only show good agreement with the measurements but also manage to capture the extreme freak wave elevation. The model thus enables the simulation of the nonlinear free surface between water and air, which is required in many applications. For example, future applications include the improvement of previous simulations of wave propagation in a Hele-Shaw cell [79] by replacing the shallow-water waves with nonlinear potential-flow waves. The model will also be used to accurately predict the location of a freak wave and its impact on a marine structure that may be subsequently placed in the basin.

Second, the variational approach has ensured ***stability*** of the simulations, both in the non-autonomous and autonomous cases. In the former, the energy of the wavemaker is accurately transferred to the waves. In the latter, when the wavemaker is turned off, the overall-energy is conserved, and its oscillations, resulting from the Hamiltonian system, are bounded with negligible amplitude that decreases with the time step. This achievement allows stable simulations of extreme waves and accurate estimation of their energy, thereby enabling a better analysis of the wave's load on marine structures.

Third, both the model and the discretisation methods were designed to be re-usable for various applications, and thus to offer ***flexibility*** to the user on many aspects: the user can change the type of finite-element expansions in the horizontal (Firedrake offers a wide range of possibilities: Galerkin, Lagrange *etc.*) and in the vertical (the Lagrange interpolation (3.21) may be replaced by any other polynomials, such as a Spline interpolation), the distribution of the vertical extension (the linear expansion (3.22) may for example be replaced to exponential distribution), the length of the domain to transform in x, by changing L_w, *etc.* Moreover, the comparison between the 1^{st}–order symplectic Euler and 2^{nd}–order Störmer-Verlet time-integration schemes assists the user in an optimal choice of temporal discretisation: while the Störmer-Verlet scheme minimizes the numerical error, it is also more time-consuming than the symplectic-Euler scheme and hence the choice should be oriented between optimised computational speed or optimised accuracy. Other higher-order schemes, such as the third-order scheme obtained variationally in [53] may be implemented for future extension.

Last but not least, the optimisation process has considerably increased the ***computational speed*** through preconditioning and careful choice of diagonalisation methods. Moreover, Firedrake codes may be run in parallel without requiring any adaptation, which, for a large number of elements (> 10000), increases the computational speed even more. Current existing wave models for industrial applications are based on large-scale computations that are time- and energy-consuming; too costly to be used as frequently as required. The present model may be used to optimise larger, industrial-scale maritime simulations, thereby saving substantial computational resources and so enhancing the overall project efficiency and quality.

Some of the possible extensions of interest to the maritime industry are presented hereafter.

3.9.2 Extention to wave-structure interactions

In section 3.8.2, it has been possible to generate a freak wave at a specific location in the basin using the dispersion effect. More generally, the fast computational-speed of the model enables the tuning of the wavemaker motion in order to adjust the location of the freak wave. Placing a maritime structure in the target area, both in the numerical and experimental wave tanks, will enable measurement of a freak wave's impact upon a vessel or a wind turbine in order to assist engineers in the design of more robust structures.

The discretisation strategies were designed with future extension to wave-structure interactions in mind. First, the restriction of the x–transform near the wavemaker allows any marine structure to be placed in the wave tank without needing to be x–transformed, which avoids additional terms in the equations. Two time schemes were proposed for better adaptability. With Störmer-Verlet, the intermediate step at time $t^{n+1/2}$ (yielding increased accuracy) nevertheless complicates the extension to wave-structure coupling. The symplectic Euler scheme, slighly less accurate but based on two full steps only, is thus an utterly viable option for wave-structure coupling.

As part of the EU Surfs-Up project, a wind turbine is currently being coupled to the present model. The previous fully linear wave-beam model developed by Salwa *et al.* [127] is extended to nonlinear potential-flow waves impacting upon a linearly vibrating turbine. When this extension materialises, modelling will be undertaken of higher and steeper waves impacting the beam, thus increasing the load and stress applied on the structure. Similarly, the wave-ship interaction model developed by Kalogirou *et al.* [78] may be extended from shallow- to deep-water nonlinear potential-flow waves to simulate higher impact on the ship.

3.9.3 Three-dimensional rogue-type wave simulation

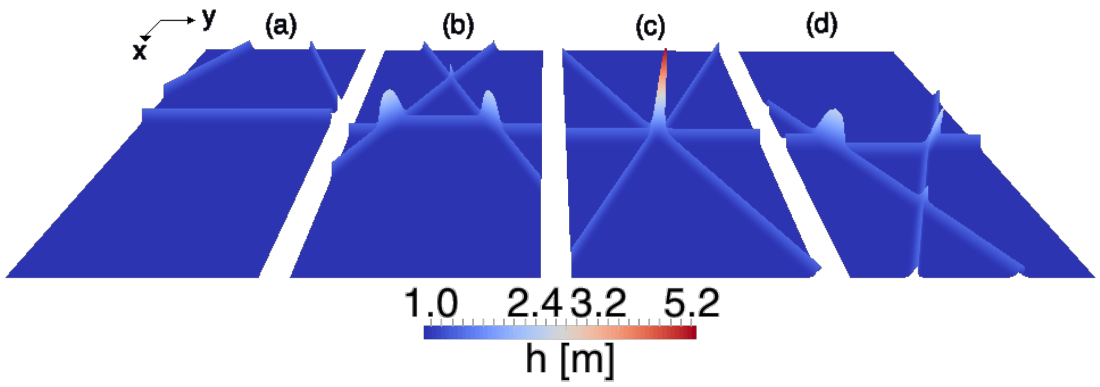


Figure 3.23: Analytical solution of a ninefold amplification resulting from the interaction of three solition, as derived by Baker [6]. (a): three initial soliton travel in the positive x direction, with amplitude $A_0 = 0.5$. (b): two-by-two soliton interactions. (c): the three solitons interact, leading to a wave of amplitude $A = 4.2 = 8.4A_0$. (d): the undisturbed solitons continue to propagate with their initial angle and amplitude.

In section 3.8.2 a two-dimensional freak wave was accurately simulated using the dispersion effect. In order to verify the ability of the present model to simulate three-dimensional Rogue-type waves, two tests are currently explored. First, the configuration used in Chapter 2 to simulate solitary-wave interaction with an oblique wall is extended by replacing the low-dispersion small-amplitude model by the present nonlinear potential-flow model. This extension requires the derivation of a soliton solution of the nonlinear potential-flow equation under the assumptions of Miles' theory. Another test case, based on the interaction of three solitons, is investigated (*cf.* Fig. 3.23). Baker [6] derived an analytical solution that may be used as initial solution of the present potential-flow model, to reach a nine-fold amplification.

3.9.4 Extension to more sophisticated wave tanks

The discretisation and implementation strategies derived and tested in this chapter may be used to develop more sophisticated wave tanks and thus offer a wide range of applications. Some suggestions of industrial interest are mentioned hereafter.

Flap-type wavemaker

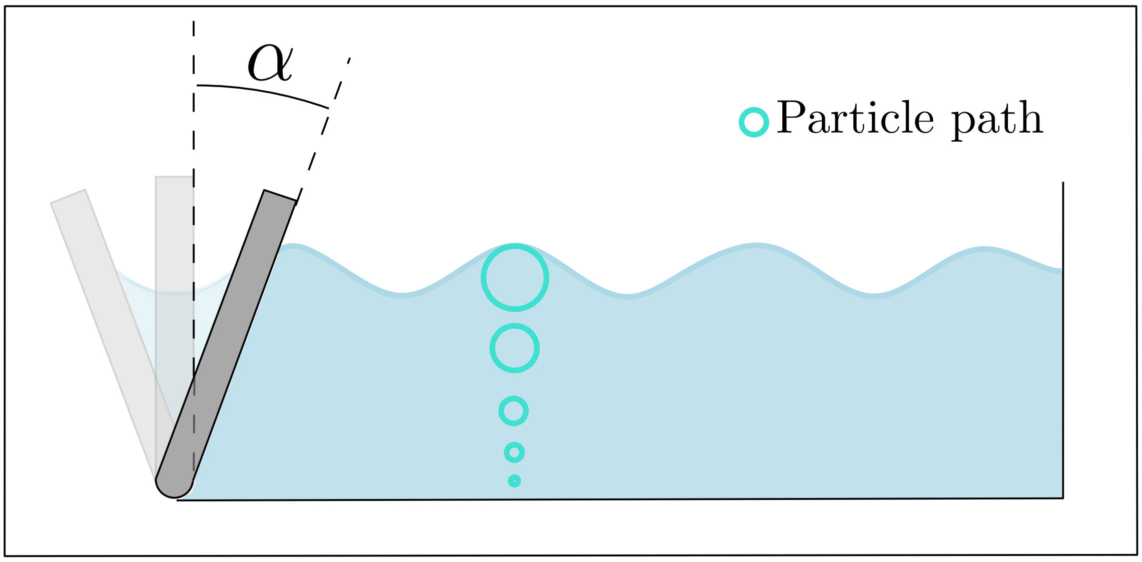


Figure 3.24: Waves generated from a flap-type wavemaker with amplitude α. The generated fluid particles follow a circular motion.

In the linear limit, fluid particles under deep-water waves follow a circular motion, whose radius decreases with depth. To mimic this fluid motion, many experimental wave tanks generate the waves via flap-type wavemakers; that is, wavemakers pivoting with an angle α from the initial vertical position (*cf.* Fig. 3.24). Extending the current piston wavemaker to a flap-type wavemaker may be achieved by following the modelling strategies, discretisation techniques and implementation method presented in this work.

Optimised wavemaker motions

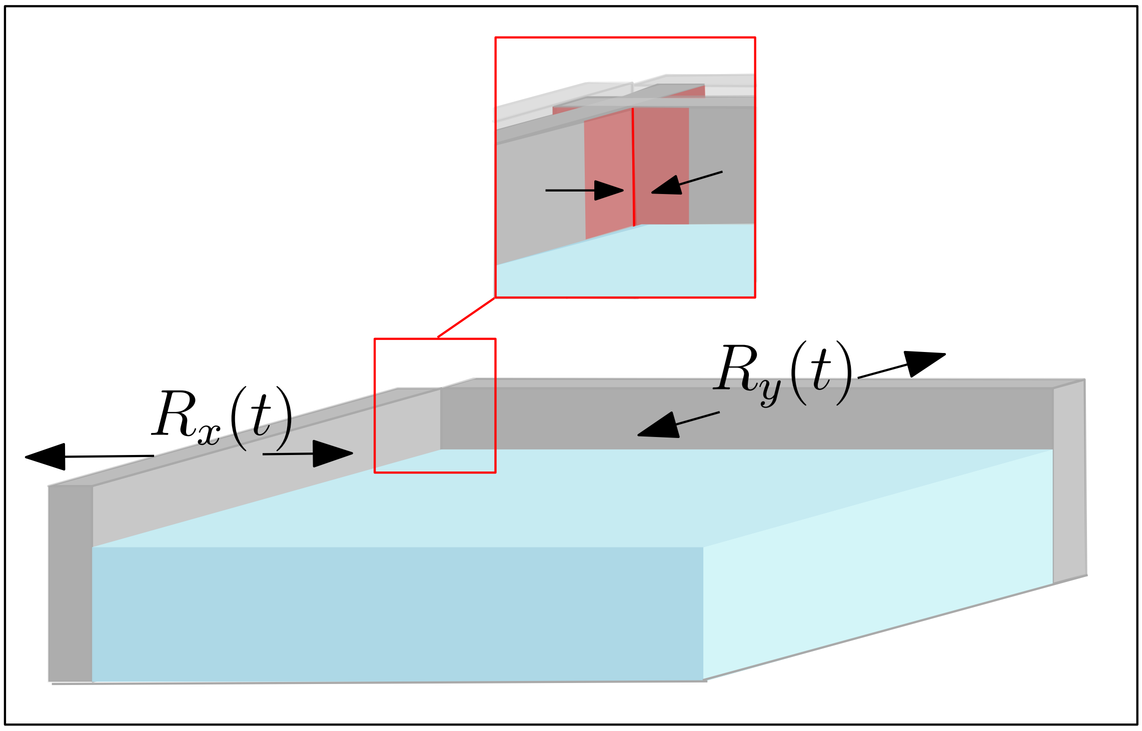


Figure 3.25: Experimental tank with two wavemakers. When they both move forward, a collision occurs in their common corner (red).

Large-scale experimental wave tanks, such as those in MARIN, sometimes contain an additional wall with wavemakers in order to generate waves in multiple direction (*cf.* Fig. 3.25). Following the methodology developed, the present model can be extended with an additional wavemaker on the wall $y = 0$, to reproduce the experimental tanks and simulate waves in several directions. The resulting configuration with two faces of wavemakers may also help the maritime industry to avoid wavemaker collision in their common corner ($x = 0$, $y = 0$) (*cf.* Fig. 3.25), by determining efficient wavemaker motions that compensate for the absence of wavemaker motion in that corner. The optimised computational speed will save a considerable amount of time and money as compared to large-scale simulations or repeated experiments.

Wave absorption at the beach

After reflection at the wall $x = L_x$, the waves travel back into the deep-water domain and adversely affect the target area. With the current model, the only way to limit disturbance of the target area by reflected wave is to increase the length of the domain so that the waves take more time to travel back to the deep-water area. This method is of course not optimal as it increases the computational time. A wave-absorbing boundary is thus required to reduce wave reflection without increase of the computation time. This can and will be done by coupling the current model to a sloping beach, partially wet and dry, with a moving dry-wet boundary, on which the waves will break and lose energy. However, the addition of a beach and the wave breaking leads to new implementation constraints that are explained and overcome in the next chapter. It will result in a complete numerical wave tank simulating the generation, the dynamics and the absorption of the waves.

Chapter 4

Numerical wave tank for offshore applications: dynamics of wavemaker, wave propagation and absorbing waves

4.1 Introduction

In Chapter 3, mathematical and numerical strategies were proposed to address key challenges in the modelling of nonlinear deep-water waves. We are now able to cost-effectively simulate the dynamic free surface of nonlinear dispersive waves. Among others advantages, the model is able to accurately capture the propagation of freak waves, which constitute a great hazard to offshore structures. Moreover, the code may be used to simulate 2D and 3D wave generation from a piston wavemaker and may therefore be used to optimise experimental set-ups in wave tanks used by the maritime industry, such as those at the Maritime Research Institute of Netherlands (MARIN).
Although faster than current CFD simulations used at MARIN (OceanWave3D, ReFRESCO), the computational speed of the model could still be improved by avoiding wave reflection on the wall opposite to the wavemaker. To this end, an absorbing boundary condition is implemented in this chapter instead of the rigid, vertical wall used in Chapter 3. The aim is to reduce the length of the numerical domain to the target area without disturbing the wave-structure-interaction tests with reflected waves that would be absent in a real sea state.

Various absorbing-boundary methods already exist to dampen water waves. A commonly used technique is the implementation of a relaxation zone (also called “forcing zone”, “damping zone”, “absorbing layer”...), in which an analytical solution of the equations is used to compute the reflection coefficient and absorb the incoming waves. This method was, for instance, implemented in the open-source library Open-Foam [143] by Jacobsen *et al.* [74], whose method defines the relaxation zone depending on the wavelength and on the geometry of the computational domain. However, the methods they provide to compute reflection factors are mainly efficient in the shallow-water regime while our numerical tank also aims to generate intermediate to deep-water waves. Peric and Maksoud [115, 116] introduced extended methods to predict the reflection coefficients before running the simulations and they obtained efficient results, including for deep-water waves. These methods provide a scaling law for adjusting the damping coefficients to the wave parameters. The relaxation method of Peric and Maksoud is, however, efficient only when an analytical solution may be estimated, typically, for regular waves, but the method is not yet applicable to irregular waves that cannot be predicted *apriori*. Our numerical tank will be used to simulate irregular sea states with unknown wave profiles at the boundary, so the above-mentioned relaxation method is not suitable. Duz *et al.* [44] provided a solution for the absorption of both regular and irregular waves based on the boundary operator initially introduced by Higdon [68, 69]. The extended boundary operator, computed from the dispersion relation for the solution of the Laplace equation and based on the angle of incidence of the generated waves, is applied on the boundary of the numerical domain through ghost cells in the mesh of the Volume-Of-Fluid numerical method. While the comparison with reference solutions showed relatively good agreement in absorbing the waves, the second-order absorbing boundary condition variant then introduced in [42] showed much improved absorption of the waves. This extended boundary operator was computed from the second-order Higdon operator to account for not only dispersive but also directional effects of the waves. However, these operators are obtained from linear wave theory and linearized Bernouilli equations, plus second-order weakly nonlinear corrections, and are currently not applicable to steep nonlinear waves such as freak waves.

In order to respond to the demands of the maritime industry, such as configuring model tests of wave-structure interactions in the basins of MARIN, we decide to dampen the water waves with a beach. With such extension of the topography, we not only provide wave absorption through shallow-water wave breaking but also design a numerical tank similar to the experimental tanks used at MARIN. However, close to the swash zone, where wave breaking occurs, the potential-flow model is not valid and the finite-element method, continuous in space, is not stable. Instead, we model the dynamics at the beach by nonlinear shallow-water equations, which we solve with a classical finite-volume method. These equations describe non-dispersive waves, but enable the modelling of wave breaking as hydraulic bores, as in Kristina *et al.* [86], that is, as mathematical discontinuity in depth and velocity. In addition, the dynamical waterline that travels along the wet/dry beach is captured through the method developed by Audusse *et al.* [5] in order to ensure non-negative water depth at the beach.

The location of the transition from deep to shallow water based on the wavelength, and therefore on the wavemaker settings, is derived in Section 4.2. The potential-flow and shallow-water equations are coupled in a variational principle from which the discrete nonlinear equations of motion are obtained. This variational approach enables us to both derive, in Section 4.3, and implement a stable numerical coupling, which ensures important conservation properties [53] and stability of the numerical scheme, with bounded energy oscillations whose amplitude tends to zero when the resolution is increased. While previous studies have led to stable coupling between deep- and shallow-water equations (see [86] for linear Boussinesq equations coupled to nonlinear shallow-water equations, and [82] for linear potential flow coupled to nonlinear shallow-water equations), the model presented here is the first fully nonlinear model that couples deep- and shallow-water equations. The nonlinear-coupling-implementation strategies are described in Section 4.4. The resulting simulations of wave generation, wave propagation and wave absorption are shown and analysed in Section 4.5. Conclusions on the efficiency of the present nonlinear numerical tank are discussed in Section 4.6.

4.2 Variational coupling of deep- and shallow-water nonlinear models

4.2.1 Definition of the domain

We simulate waves propagating in the x–direction in a two-dimensional vertical wave basin, on one side of which a time-dependent wavemaker drives the wave motion; on the other side, a beach is included to dampen the waves. The depth at rest is characterised by

$$H_r(x) = H_0 - b(x), \tag{4.4}$$

where the beach topography

$$b(x) = s_B(x - x_B)\Theta(x - x_B), \tag{4.5}$$

with Θ the Heaviside function, so that the water depth is constant for $x \leq x_B$ and decreases along the beach with slope $s_B > 0$ for $x \geq x_B$ (see Fig. 4.1). For $t > 0$, the wavemaker oscillatory motion

$$R(t) = -\gamma \cos(\omega t) \tag{4.6}$$

around the position $x = 0$, with γ its amplitude and ω its frequency, leads to a surface deviation $\eta(x,t)$ from the depth at rest $H_r(x)$, yielding a total depth $h(x,t) = H_r(x) + \eta(x,t)$. Our goal is to calculate the total depth $h(x,t)$ as well as the velocity potential $\phi(x,z,t)$.

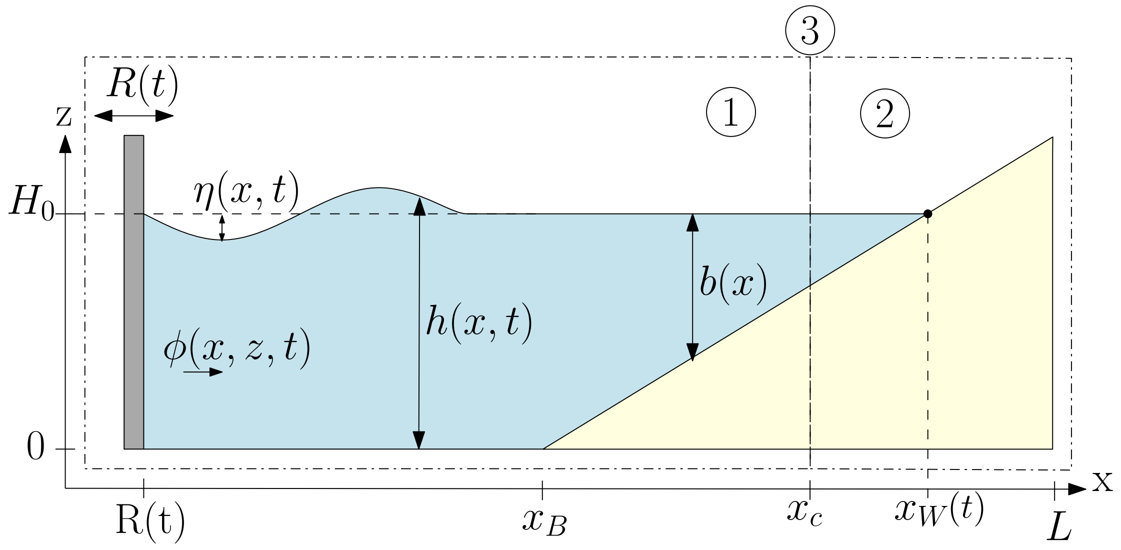


Figure 4.1: Two-dimensional vertical domain containing a piston wavemaker on the left boundary to generate the waves and a beach on the right boundary to limit their reflection. The basin is divided into two subdomains: a deep-water domain ① where nonlinear potential-flow equations are solved and a shallow-water domain ② where the nonlinear shallow-water equations are solved. At the interface ③ between the two domains, coupling conditions are derived.

Figure 4.1 shows the two-dimensional vertical domain, divided into two subdomains. In the first domain ①, which includes a piston wavemaker and seabed topography and is hereafter referenced as the "deep-water domain" $\Omega_D = \{R(t) \leq x < x_c; 0 \leq z \leq h(x,t)\}$, the wave motion and velocity may, as shown in Chapter 3, be accurately described by the nonlinear potential-flow equations. However, as waves travel along the beach in the domain ②, hereafter referenced as the "shallow-water domain" $\Omega_S = \{x_c < x \leq x_W(t)\}$, the seabed influences the wave profile and causes the waves to break on the beach. While necessary to dampen the wave energy, the breaking is modelled as a discontinuous phenomenon that cannot be captured by the classical continuous finite-element method used in the deep-water domain, and thus leads to numerical instabilities in the potential-flow model. Hence, the wave motion and velocity are instead described by the nonlinear shallow-water equations in the shallow-water domain. These equations enable the modelling of wave breaking as hydraulic bores, as in Kristina *et al.* [86], and ensure non-negative depth at the beach through the method developed by Audusse *et al.* [5]. To link the solutions from the two subdomains, a fixed coupling interface Γ_c between the deep- and shallow-water domains is set in a transition zone ③ in which both the potential-flow and shallow-water equations are valid and numerically stable. Thus, the location x_c of the transition from deep to shallow water is defined so that $h(x_c)$ is between the deep-water limit $h \geq \lambda/2$ and the shallow-water limit $h \leq \lambda/20$. The potential-flow and shallow-water equations are coupled in a variational principle as described in the next section.

4.2.2 Variational approach

We aim to understand and to simulate the dynamics occurring in the deep- and shallow-water domains including the dynamic coupling at $x = x_c$. The evolution of the water-depth $h(x,t)$ and velocity potential $\phi(x,z,t)$ in the numerical basin of Fig. 4.1 are described by Luke's [92] variational principle for nonlinear waves

$$0 = \delta \int_0^T \int_{R(t)}^{x_W(t)} \int_0^{h(x,t)} \left\{ \partial_t \phi + \frac{1}{2}(\nabla\phi)^2 + g\,(z - H_r(x)) \right\} \mathrm{d}x\,\mathrm{d}z\,\mathrm{d}t. \tag{4.7}$$

As explained in section 4.2.1, the domain is split into a deep-water domain Ω_D and a shallow-water domain Ω_S connected by the interface Γ_c at the coupling position $x = x_c$. As a consequence, the variational principle (4.7) may be split into the two sub-domains as

$$
\begin{aligned}
0 = \delta \int_0^T \Bigg\{ & \int_{\Omega_D} \left[\partial_t \phi + \frac{1}{2} (\nabla \phi)^2 + g\,(z - H_r(x)) \right] \mathrm{d}x\,\mathrm{d}z \\
& + \int_{\Omega_S} \left[h \partial_t \check{\phi} + \frac{1}{2} h (\partial_x \check{\phi})^2 + gh \left(\frac{1}{2} h - H_r(x) \right) \right] \mathrm{d}x \Bigg\} \,\mathrm{d}t,
\end{aligned}
\tag{4.8}
$$

where the velocity potential has been depth-averaged in the one-dimensional shallow-water domain Ω_S with additional simplifications as

$$
\check{\phi}(x,t) = \frac{1}{h} \int_0^h \phi(x,z,t)\,\mathrm{d}z. \tag{4.9}
$$

Variations of h, ϕ, and $\check{\phi}$ in the variational principle (4.8) with end-point conditions $\delta\phi(0) = \delta\phi(T) = 0$ and $\delta\check{\phi}(0) = \delta\check{\phi}(T) = 0$ lead to the following:

$$
\begin{aligned}
\int_0^T \Bigg\{ \int_R^{x_c} \Bigg[& \int_0^h \delta\phi \left[-\partial_{xx}\phi - \partial_{zz}\phi \right] \mathrm{d}z \\
& + \delta h \left(g(h - H_r) + \partial_t \phi + \frac{1}{2} (\nabla\phi)^2 \right) |_{z=h} \\
& - \delta\phi \left(\partial_z \phi \right) |_{z=0} + \delta\phi \left(-\partial_t h - \partial_x h \partial_x \phi + \partial_z \phi \right) |_{z=h} \Bigg] \mathrm{d}x \\
+ \int_{x_c}^{x_W(t)} & \left[\delta h \left(\partial_t \check{\phi} + \frac{1}{2} (\partial_x \check{\phi})^2 + g(h - H_r) \right) \right. \\
& \left. + \delta\check{\phi} \left(-\partial_t h - \partial_x \check{\phi} \partial_x h - h \partial_{xx} \check{\phi} \right) \right] \mathrm{d}x \\
+ \Bigg[\int_0^h & \delta\phi \left(\dot{R} - \partial_x \phi \right) \mathrm{d}z \Bigg]_{x=R} + \left[\delta\check{\phi} \left(h \partial_x \check{\phi} - h \dot{x}_W \right) \right]_{x=x_W} \\
+ \Bigg[\int_0^h & \delta\phi \left(\partial_x \phi \right) \mathrm{d}z \Bigg]_{x=x_c} - \left[\delta\check{\phi} \left(h \partial_x \check{\phi} \right) \right]_{x=x_c} \\
+ \Bigg[\delta x_W & \left(h \partial_t \check{\phi} + \frac{1}{2} h (\partial_x \check{\phi})^2 + gh \left(\frac{1}{2} h - H_r(x) \right) \right) \Bigg]_{x=x_W} \Bigg\} \,\mathrm{d}t = 0,
\end{aligned}
\tag{4.10}
$$

where $\dot{()}$ denotes the time derivative. By definition, the water depth at the waterline $x = x_W$ is $h(x_W, t) = 0$. Therefore, the terms of Eq. (4.10) evaluated at $x = x_W$ vanish. Arbitrariness of $\delta\phi$, δh and $\delta\check{\phi}$ in the resulting variational principle leads to the nonlinear potential-flow equations in deep water:

$$\delta\phi: \quad \partial_{xx}\phi + \partial_{zz}\phi = 0, \qquad \text{in } \Omega_D, \tag{4.11a}$$

$$\delta h: \quad \partial_t\phi + \frac{1}{2}(\nabla\phi)^2 + g(h - H_r) = 0, \qquad \text{at } z = h, \tag{4.11b}$$

$$\delta\phi_{z=h}: \quad \partial_t h + \partial_x h \partial_x \phi - \partial_z \phi = 0, \qquad \text{at } z = h, \tag{4.11c}$$

$$\delta\phi_{z=0}: \quad \partial_z\phi = 0, \qquad \text{at } z = 0, \tag{4.11d}$$

$$\delta\phi_{x=R}: \quad \partial_x\phi = \dot{R}, \qquad \text{at } x = R, \tag{4.11e}$$

as well as the nonlinear potential-flow equations in shallow water:

$$\delta\check{\phi}: \quad \partial_t h + \partial_x h \partial_x \check{\phi} + h\partial_{xx}\check{\phi} = 0, \qquad \text{in } \Omega_S, \tag{4.12a}$$

$$\delta h: \quad \partial_t\check{\phi} + \frac{1}{2}(\partial_x\check{\phi})^2 + g(h - H_r) = 0, \qquad \text{in } \Omega_s, \tag{4.12b}$$

or the equivalent nonlinear shallow-water equations in terms of h and $hu = h\partial_x\check{\phi}$:

$$\partial_t h + \partial_x(hu) = 0, \qquad \text{in } \Omega_S, \tag{4.13a}$$

$$\partial_t(hu) + \frac{1}{2}hu^2 + \frac{1}{2}g\partial_x(h^2) = g\partial_x H_r, \qquad \text{in } \Omega_s. \tag{4.13b}$$

In the shallow-water regime, the phase speed depends on the wave height, meaning that the crests of the waves travel faster than their troughs (as can be shown with the dispersion relation). As a consequence, the left and right speeds of the wave are different which leads to an increase in the wave height: kinematic energy is transformed into potential energy. When the wave crest reaches a critical height, the steepness of the wave becomes too large and wave breaking occurs: energy is then lost through viscous dissipation. In this chapter, the turbulence of breaking waves is modelled through hydraulic bores which consist in mathematical discontinuity in the fluid depth and velocity. Depending on the left and right speeds and depths of the bore, the hydraulic jump can evolve into a rarefaction or a shock wave, for which analytical solutions can be derived by solving the Riemann problem. The energy lost during wave breaking can be estimated analytically in terms of the energy density, the energy flux and the speed of the bore from the shallow-water equations (4.13). When multiplied by the water depth h, the shallow-water equations (4.13) can indeed be written in a form that conserves mass and momentum but in which energy is lost in the case of an hydraulic jump (see, *e.g.*, [119]).

The deep- and shallow-water domains are continuously connected through $\Gamma_c = \{x = x_c\}$ meaning that the boundary conditions emerging from the variations at x_c need to be addressed simulatenously to yield the coupling conditions. This is done in the next section.

4.2.3 Coupling mechanisms at the interface between deep and shallow water

Since physical information is transferred bidirectionally from deep to shallow water and from shallow to deep water, a coupling condition must be derived. Following Klaver *et al.* [82], setting and substituting

$$\left(\delta\check{\phi}\right)_{x=x_c} = \left(\frac{1}{h}\int_0^h \delta\phi \, \mathrm{d}z\right)_{x=x_c}, \tag{4.14}$$

into the variational principle (4.10) and using arbitrariness of $\delta\phi$ at $x = x_c$ lead to the coupling boundary condition for the deep-water equations:

$$(h\partial_x\phi)_{x=x_c} = \left(h\partial_x\check{\phi}\right)_{x=x_c} = (hu)_{x=x_c} \qquad \text{for } 0 \le z \le h \text{ and } x = x_c. \tag{4.15}$$

On the other hand, setting and substituting

$$(\delta\phi)_{x=x_c} = \left(\delta\check{\phi}\right)_{x=x_c}, \tag{4.16}$$

into the variational principle (4.10) and using the arbitrariness of $\delta\check{\phi}$ at $x = x_c$ lead to the coupling boundary condition for the shallow-water equations:

$$\left(h\partial_x\check{\phi}\right)_{x=x_c} = \left(\int_0^h \partial_x\phi \, \mathrm{d}z\right)_{x=x_c} \qquad \text{at } x = x_c, \tag{4.17}$$

which, in terms of h and hu, reads

$$(hu)_{x=x_c} = \left(\int_0^h \partial_x\phi\right)_{x=x_c} \mathrm{d}z \qquad \text{at } x = x_c. \tag{4.18}$$

We therefore need to solve the deep-water potential-flow equations (4.11a)-(4.11e) together with the nonlinear shallow-water equations (4.13a)-(4.11b) and coupling conditions Eq. (4.15) and Eq. (4.18). The next section explains the methodology used to solve and couple the two sets of equations numerically.

4.3 Numerical coupling of deep- and shallow-water domains

In this section, numerical strategies are derived to both discretize and couple the nonlinear potential-flow and nonlinear shallow-water equations. This numerical coupling involves addressing several challenges, including: dealing with a moving wavemaker, free-surface and waterline boundaries; handling the breaking waves at the beach; and, ensuring consistent and accurate transfer of information through the nonlinear coupling interface between the deep- and shallow-water domains. The spatial and temporal discretisation methods and the implementation strategies used to tackle these challenges are now considered.

4.3.1 Spatial coupling strategies

Discretization of the domain

The numerical domain as defined by $\Omega_D \cup \Omega_S$ admits three time-dependent boundary conditions: one at the wavemaker $x = R(t)$, one at the free surface $z = h(x,t)$ and one at the waterline $x = x_W(t)$. While the wavemaker motion is prescribed, the other two boundary conditions are part of the solution and are thus not known *a-priori*. As a consequence, implementing a time-dependent mesh that follows the boundaries would require an iterative process, which is a potentially costly approach that we choose to avoid. In Chapter 3, a solution was proposed based on the transformation of the domain in x and z as follows:

$$x \quad \rightarrow \quad \hat{x} = \frac{x - \tilde{R}}{L_w - \tilde{R}} L_w \qquad \text{and} \qquad z \quad \rightarrow \quad \hat{z} = z\frac{H_0}{h(x,t)}, \tag{4.19}$$

with $L_w = O(\lambda)$ and

$$\tilde{R}(x,t) = R(t)\Theta(L_w - x) = \begin{cases} R(t), & \text{if } x \leq L_w, \\ 0, & \text{if } x > L_w. \end{cases} \tag{4.20}$$

As this transformation was shown to be efficient in Chapter 3, the same method is applied to the deep-water subdomain of the present numerical tank, which thus becomes

$$\hat{\Omega}_D = \{0 \leq \hat{x} \leq x_c; 0 \leq \hat{z} \leq H_0\}. \tag{4.21}$$

For clarity, all hats are subsequently omitted. Figure 4.2 shows the deep-water numerical domain. In shallow water, the solutions being defined at the free surface, only the boundaries at the coupling ($x = x_c$) and waterline ($x = x_W$) coordinates must be considered. While x_c is fixed, the waterline coordinate x_W moves along the beach and must be estimated together with the free-surface solution to ensure non-negative depth for $x > x_W(t)$. To avoid mesh motion following the boundary $x_W(t)$, we set a new boundary at $x = L > \max_t(x_W(t))$ (*cf.* Fig. 4.1), so that the equations are solved in a fixed numerical domain. Moreover, (4.4) and (4.5) together imply that for $x \geq x_c$,

$$\begin{aligned} H_r(x) &= H_0 - s_B(x - x_B) \\ &= H_0 - s_B(x - x_c + x_c - x_B) \\ &= H_r(x_c) - s_B(x - x_c). \end{aligned} \tag{4.22}$$

We thus introduce the shallow-water depth at rest and beach topography as

$$\check{H}_r(x) \equiv H_r(x_c) - \check{b}(x), \qquad \text{with} \qquad \check{b}(x) = s_B(x - x_c). \tag{4.23}$$

Figure 4.2 illustrates the definition of $\check{b}(x)$ and $\check{H}_r(x)$ in the shallow-water numerical domain. More details on how to accommodate the discrete waterline with the Audusse method are given in the next paragraph.

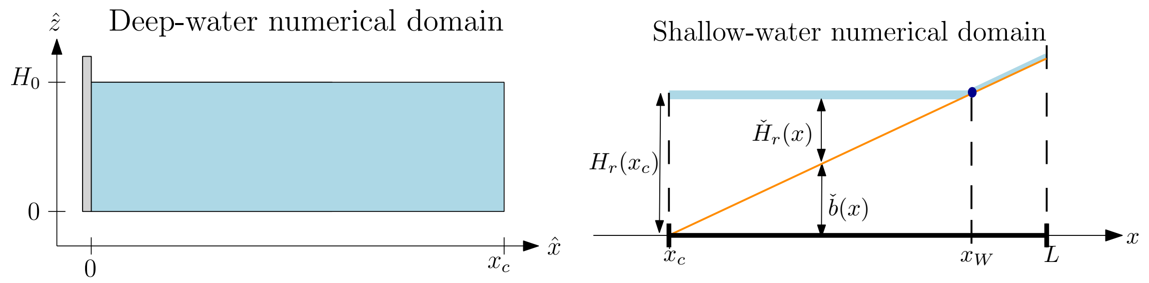


Figure 4.2: Left: Fixed deep-water numerical domain (blue) as defined by $\hat{\Omega}_D$. Right: Fixed shallow-water numerical domain as defined by $\check{\Omega}_S$ (bold black) and corresponding water depth at rest $\check{H}(\check{x})$ (blue) and beach topography $\check{b}(\check{x})$ (orange).

Discretization methods

The numerical domain is divided into two different meshes: a transformed, fixed 2D mesh for the deep-water domain and a fixed 1D mesh for the shallow-water domain. In the deep-water domain, the nonlinear potential-flow equations are solved with the finite-element method. As in Chapter 3, the domain is split into N_x elements of depth H_0 and length Δx_{DW} that must satisfy $\Delta x_{DW} < \lambda$ to ensure that waves are consistently resolved. To distinguish the surface and interior velocity-potential evaluations, the velocity potential $\phi(x, z, t)$ is expanded along the depth of each element with Lagrange polynomials $\tilde{\varphi}(z)$ of order n_z, so that

$$\begin{aligned}\phi(x,z,t) &= \psi_i(x,t)\tilde{\varphi}_i(z),\\ &= \psi_1(x,t)\tilde{\varphi}_1(z) + \psi_{i'}(x,t)\tilde{\varphi}_{i'}(z),\\ &= \begin{cases}\psi_1(x,t) & \text{at } z = H_0,\\ \psi_{i'}(x,t) & \text{at } z = z_{i'} < H_0,\end{cases}\end{aligned} \tag{4.24}$$

with $i \in [1, n_z + 1]$ and $i' \in [2, n_z + 1]$. Transformation of the variational principle (4.8) with (4.19) and substitution of the Lagrange expansions (4.24) enable us to express all of the z–integrals within constant matrix coefficients, thereby reducing the original two-dimensional variational principle to a one-dimensional horizontal variational principle with only x–dependent functions: the deep-water surface and interior velocity potentials $\psi_1(x, t)$ and $\psi_{i'}(x, t)$ respectively; the depth-averaged shallow-water velocity potential $\check{\phi}$; and, the depth $h(x, t)$. The horizontal deep-water coefficients $\psi_1(x, t)$, $\psi_{i'}(x, t)$ and $h(x, t)$ are then interpolated with first-order continuous Galerkin expansions in the horizontal, as

$$\begin{aligned}\psi_1(x,t) &= \psi_{1p}(t)\varphi_p(x),\\ \psi_{i'}(x,t) &= \psi_{i'p}(t)\varphi_p(x),\\ h(x,t) &= h_p(t)\varphi_p(x),\end{aligned} \tag{4.25}$$

where $\varphi_p(x)$ are the basis functions on the deep-water horizontal elements, defined for $x \in [0, x_c]$ and $p \in [0, c^-]$, with 0 the node at $x = 0$ and c^- the coupling node at $x = x_c$. A scheme of the corresponding mesh is given in Fig. 4.3. Substitution of the finite element interpolations (4.25) into the continuous horizontal variational principle leads to the discrete variational principle (4.39) given in section 4.3.1. For clarity, the surface and interior velocity potentials are merged into ψ_i with $i \in [1, n_z + 1]$. In section 4.3.1, we show that variations of (4.39) with respect to ψ_i and h lead to the weak formulations of the discrete, transformed potential-flow equations with a coupling condition at $x = x_c$.

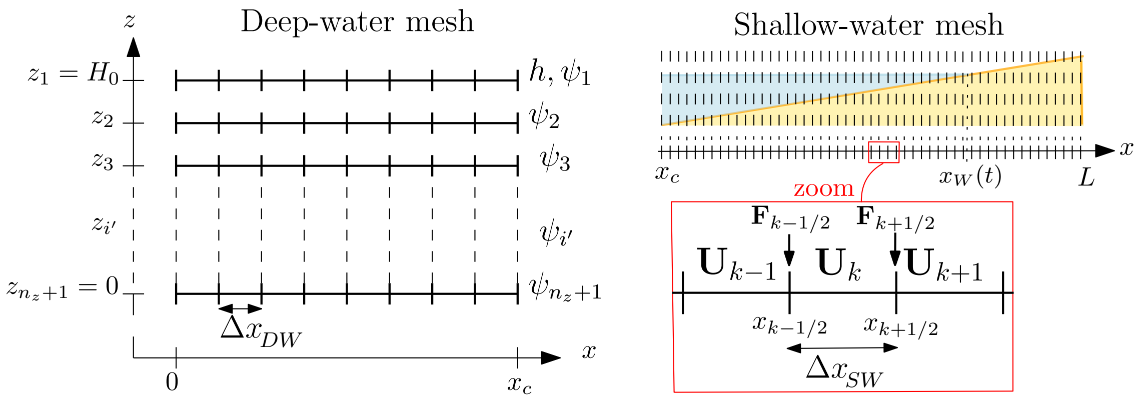


Figure 4.3: Left: deep-water mesh which includes $n_z + 1$ horizontal layers, each containing N_x elements of size Δx_{DW}. Right: shallow-water mesh that contains $\check{N}_x$ volumes of length Δx_{SW} and $\check{N}_x + 1$ interfaces on which the inward and outward fluxes of each volume are determined.

In shallow water, near the breaking zone, the finite-element approach used in the deep-water domain, which is based on continuous expansions, would not be stable due to the increasing steepness of the breaking waves. Instead, we implement a Godunov finite-volume method that can accommodate breaking waves, as in Kristina *et al.* [86]. To facilitate hydraulic bores, the system of nonlinear shallow-water Eqs. (4.13a-b) is written in conservative form as

$$\partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) = \mathbf{S}, \tag{4.26}$$

with the vector $\mathbf{U}$, flux $\mathbf{F}(\mathbf{U})$ and supplementary term $\mathbf{S}$, as follows

$$\mathbf{U} = (h, hu)^T, \quad \mathbf{F}(\mathbf{U}) = \left(hu, hu^2 + \frac{1}{2}gh^2\right)^T \quad \text{and} \quad \mathbf{S} \; = \; \left(0, ghd_x\check{H}_r\right)^T. \tag{4.27}$$

The spatial domain is discretised with $\check{N}_x$ cells $X_k \; = \; [x_{k-1/2}, x_{k+1/2}]$ of length Δx_{SW}, on each of which the solutions $\mathbf{U}$ are averaged, yielding the discrete solutions $\mathbf{U}_k$ defined as

$$\mathbf{U}_k(t) = \frac{1}{\Delta x_{SW}} \int_{x_{k-1/2}}^{x_{k+1/2}} \mathbf{U}(x,t)\, \mathrm{d}x. \tag{4.28}$$

Since the basis functions associated to $\mathbf{U}_k$ are discontinuous and only taken to be C^0 as piecewise constant within a volume and since those associated with ψ_i are C^1 continuous across nodes and approximated linearly between nodes,

$$\Delta x_{SW} \propto (\Delta x_{DW})^2 . \tag{4.29}$$

This ensures a consistent matching between the leading order finite-volume method and the first order finite-element method. A particular advantage of the finite-volume method is that it also estimates the flux at the interface between each cell, thereby allowing the modelling of discontinuous waves. The discrete flux

$$\mathbf{F}_{k\pm1/2}(t) = \mathbf{F}(\mathbf{U}(x_{k\pm1/2}, t)), \tag{4.30}$$

evaluated at the interface $x = x_{k\pm1/2}$, is computed through the Harten-Lax-van Leer (HLL) flux [64], which is defined in Appendix C.1. Figure 4.3 shows the mesh and indicates the corresponding state and flux evaluations. The topographic source term $\mathbf{S}$ is evaluated by assuming hydrostatic flow with $u \ll \sqrt{gh}$ and a balance between momentum flux and momentum source terms (see [5]), yielding

$$\mathbf{S} = (0, ghd_x\check{H}_r) = \left(0, \partial_x\left(g\frac{1}{2}h^2\right)\right).$$

This ensures that hydrostatic flow at rest stays at rest numerically. Integration of $\mathbf{S}$ over a volume X_k yields the interpolation of the discretised source term $\mathbf{S}_k$ as

$$\mathbf{S}_k = \left(0, \int_{x_{k-1/2}}^{x_{k+1/2}} gh(x,t)d_x\check{H}_r \, \mathrm{d}x\right) = \left(0, \frac{1}{2}gh_{k+1/2^-}^2 - \frac{1}{2}gh_{k-1/2^+}^2\right), \tag{4.31}$$

where $h_{k+1/2^-}$ and $h_{k-1/2^+}$ are obtained from the Audusse method [5] to ensure non-negative depth at the waterline through

$$h_{(k+1/2)^-} = \max(h_k + \check{b}_k - \check{b}_{k+1/2}, 0), \tag{4.32a}$$

$$h_{(k-1/2)^+} = \max(h_k + \check{b}_k - \check{b}_{k-1/2}, 0), \tag{4.32b}$$

$$\check{b}_{k+1/2} = \max(\check{b}_k, \check{b}_{k+1}), \tag{4.32c}$$

with $\check{b}_k$ the discrete beach topography defined in (4.23). The above method developed by Audusse *et al.* [5] ensures stability of the simulations despite the time-dependent waterline motion at

$x = x_w(t) < L$. Integration of the conservative form (4.26) over one cell X_k leads to the space-discrete nonlinear shallow-water equations given by

$$\dot{\mathbf{U}}_k(t) + \frac{1}{\Delta x_{SW}}\left(\mathbf{F}_{k+1/2}(t) - \mathbf{F}_{k-1/2}(t)\right) = \mathbf{S}_k, \tag{4.33}$$

where a dot superscript denotes a time derivative. Distinguishing the first and second components of $\boldsymbol{F}(\boldsymbol{U})$ and $\boldsymbol{S}(\boldsymbol{U})$ as

$$\begin{cases} F^h_{k\pm1/2}(\boldsymbol{U}) = [hu]_{k\pm1/2}\,, \\ F^{hu}_{k\pm1/2}(\boldsymbol{U}) = \left[\dfrac{(hu)^2}{h} + \dfrac{1}{2}gh^2\right]_{k\pm1/2}, \end{cases} \tag{4.34}$$

$$\text{and} \quad \begin{cases} S^h_k(\boldsymbol{U}) = 0, \\ S^{hu}_k(\boldsymbol{U}) = \dfrac{1}{2}gh^2_{k+1/2} - \dfrac{1}{2}gh^2_{k-1/2}, \end{cases} \tag{4.35}$$

Eq. (4.33) may be split into

$$\dot{h}_k(t) = -\frac{1}{\Delta x_{SW}}\left(F^h_{k+1/2}(t) - F^h_{k-1/2}(t)\right), \tag{4.36a}$$

$$\dot{hu}_k(t) = -\frac{1}{\Delta x_{SW}}\left(F^{hu}_{k+1/2}(t) - F^{hu}_{k-1/2}(t)\right) + \mathbf{S}^{hu}_k. \tag{4.36b}$$

Space-discrete coupled variational principle

The averaged shallow-water velocity u_k satisfies

$$\begin{aligned} u_k =& \frac{1}{\Delta x_{SW}} \int_{x_{k-1/2}}^{x_{k+1/2}} u(x,t)\,\mathrm{d}x \\ =& \frac{1}{\Delta x_{SW}} \int_{x_{k-1/2}}^{x_{k+1/2}} \partial_x \check{\phi}(x,t)\,\mathrm{d}x \\ =& \frac{1}{\Delta x_{SW}} \left[\check{\phi}(x_{k+1/2},t) - \check{\phi}(x_{k-1/2},t)\right]. \end{aligned} \tag{4.37}$$

The discrete C^0 shallow-water velocity may thus be expressed in terms of the discrete C^1 shallow-water velocity potential $\check{\phi}_h$ evaluated at the cell interfaces through

$$\check{\phi}_h = \phi_l(t)\varphi_l(x), \quad \text{for } l \in [c^+, c + \check{N}_x], \tag{4.38}$$

with $\varphi_l(x)$ the 1st–order basis function that satisfies $\varphi_l(x_k) = \delta_{kl}$. Transformation of the deep-water domain with (4.19) and substitution of the deep-water vertical (Lagrange) and horizontal (finite-element) expansions (4.24) and (4.25) and shallow-water expansions (4.38) lead to the space-discrete variational principle as

$$\begin{aligned}0 = \delta \int_0^T \frac{1}{2H_0}&\left[h_l A_{lqm}\psi_{iq}\psi_{jm}\tilde{M}_{ij} + \frac{h_l h_n}{h_p}\psi_{iq}\tilde{S}_{ij}\psi_{jm}E_{pmqln} - 2A_{mlq}h_l\psi_{iq}\tilde{D}_{ij}\psi_{jm}\right] \\ &+\frac{H_0}{2h_l}J_{lmq}\psi_{iq}\tilde{A}_{ij}\psi_{jm} + gh_l(\frac{1}{2}h_p M_{pl} - HI_l) - \psi_{1q}\left[N_{ql}\frac{h_l}{L_w} + M_{ql}\partial_t h_l\right] \\ &+\frac{1}{H_0}\partial_t R\, h_0\, \psi_{i0}I_i + \left[M_{kr}h_k\partial_t\phi_r + \frac{1}{2}h_k\phi_r\phi_s A_{krs} + gh_k(\frac{1}{2}h_r M_{kr} - \check{H}I_k)\right]\,\mathrm{d}t,\end{aligned} \tag{4.39}$$

with $\varphi_{l,m,n,p,q}$ defined in $[0, x_c]$, and $\varphi_{k,r,s}$ defined in $[x_c, L]$. The left and right limits of node c at $x = x_c$ are used to highlight the fact that the basis function at $x = x_c$ is either the deep-water value φ_{c^-}, which is not defined for $x > x_c$, or the shallow-water value φ_{c^+} which is not defined for $x < x_c$. The spatial matrices involved in (4.39) are defined in Appendix C.2. The space-discrete coupling conditions are derived from this variational principle in the next section.

Discrete coupling conditions

At the coupling node $x = x_c$, the shallow-water potential is the depth-averaged deep-water velocity potential, *i.e.*

$$\phi_c = \frac{1}{h}\int_0^h \phi\,\mathrm{d}z = \frac{1}{H_0}\psi_{ic}\tilde{I}_i. \tag{4.40}$$

Therefore, the shallow-water expansions may be split into

$$\phi = \phi_c\varphi_{c^+} + \phi_{k'}\varphi_{k'} = \frac{1}{H_0}\psi_{ic}\tilde{I}_i\varphi_{c^+} + \phi_{k'}\varphi_{k'}, \tag{4.41}$$

$$h = h_c\varphi_{c^+} + h_{k'}\varphi_{k'}, \tag{4.42}$$

in which $k' \in]c^+, \check{N}_x + c]$. As the deep-water model is solved with Firedrake, in which the spatial dependency is implemented continuously, the deep-water part of the variational principle (4.39) is written in x–continuous form. Substitution of (4.41) leads to

$$
\begin{aligned}
\delta \int_0^T \int_0^{x_c} \Bigg\{ & \frac{1}{2\Upsilon}\left[h\partial_x\psi_i\tilde{M}_{ij}\partial_x\psi_j + \frac{1}{h}(\partial_x h)^2\psi_i\tilde{S}_{ij}\psi_j - 2\partial_x h\partial_x\psi_i\tilde{D}_{ij}\psi_j\right] \\
& + \Upsilon\left[\frac{1}{2h}\psi_i\tilde{A}_{ij}\psi_j + \frac{1}{H_0}\left(gh(\frac{1}{2}h - H) - \psi_1\partial_t h\right)\right] \\
& - \psi_1\frac{(x-L_w)}{L_w}\partial_t\tilde{R}\partial_x h\Bigg\}\,\mathrm{d}x + \left(\frac{1}{H_0}\partial_t R h\psi_i\tilde{I}_i\right)_{x=0} \\
& + \frac{1}{H_0}\left(h_c M_{c^+c^+} + h_{k'}M_{k'c^+}\right)\tilde{I}_i d_t\psi_{ic} + \left(h_c M_{c^+k'} + h_{r'}M_{r'k'}\right)d_t\phi_{k'} \\
& + \frac{1}{H_0}\phi_{k'}\tilde{I}_i\psi_{ic}\left[h_c A_{c^+k'c^+} + h_{k'}A_{k'k'c^+}\right] \\
& + \frac{1}{2H_0^2}\psi_{ic}\tilde{I}_i\psi_{jc}\tilde{I}_j\left[h_c A_{c^+c^+c^+}) + h_{k'}A_{k'c^+c^+}\right] \\
& + g\Bigg[\frac{1}{2}\left(h_c M_{c^+c^+}h_c + h_{k'}M_{k'r'}h_{r'}\right) \\
& \qquad + h_c M_{c^+k'}h_{k'} - H\left(h_c I_{c^+} + h_{k'}I_{k'}\right)\Bigg]\,\mathrm{d}t = 0,
\end{aligned}
\tag{4.43}
$$

with $\Upsilon = WH_0/L_w$. To obtain the deep-water coupling condition, variations of (4.43) with respect to ψ, ϕ and h are taken, keeping in mind that the variations of ψ_{ic} in (4.41) must be transformed with (4.19), leading to

$$
\delta\phi = \frac{\delta\psi_{ic}}{H_0}\tilde{I}_i\varphi_{c^+} - \frac{\delta h_c}{H_0 h_c}\psi_{ic}\tilde{G}_i\varphi_{c^+} + \phi_{k'}\varphi_{k'}, \tag{4.44}
$$

$$
\delta h = \delta h_c\varphi_{c^+} + \delta h_{k'}\varphi_{k'}, \tag{4.45}
$$

with

$$
\tilde{G}_i = \int_0^{H_0} z\partial_z\tilde{\varphi}_i\,\mathrm{d}z. \tag{4.46}
$$

The variations of (4.43) thus yield

$$\begin{aligned}
\int_0^T \int_0^{x_c} & \delta\psi_{iq}\Bigg\{ \frac{1}{\Upsilon}\Bigg[h\partial_x\varphi_q\partial_x\psi_j\tilde{M}_{ij} + \frac{1}{h}(\partial_x h)^2\psi_j\varphi_q\tilde{S}_{ij} \\
&\quad - \partial_x h\left(\partial_x\varphi_q\tilde{D}_{ij}\psi_j + \varphi_q\partial_x\psi_j\tilde{D}_{ji}\right)\Bigg] + \frac{\delta_{q0}}{H_0}\varphi_0\partial_t R h_0\tilde{I}_i \\
&\quad +\Upsilon\left[\frac{H_0}{h}\varphi_q\psi_j\tilde{A}_{ij} - \frac{\delta_{i1}}{H_0}\varphi_q\left[\partial_t h + \frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x h\right]\right]\Bigg\} \\
& +\delta h_l\Bigg\{\frac{1}{2\Upsilon}\Bigg[\varphi_l\partial_x\psi_i\tilde{M}_{ij}\partial_x\psi_j + \left(-\frac{\varphi_l}{h^2}(\partial_x h)^2 + \frac{2}{h}\partial_x h\partial_x\varphi_l\right)\psi_i\tilde{S}_{ij}\psi_j \\
&\quad - 2\partial_x\varphi_l\partial_x\psi_i\tilde{D}_{ij}\psi_j\Bigg] + \frac{\Upsilon}{H_0}\left[\varphi_l g\,(h-H) - \psi_1\frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x\varphi_l\right] \\
&\quad -\frac{\Upsilon}{2h^2}\varphi_l\psi_i\tilde{A}_{ij}\psi_j + \delta_{l0}\frac{1}{H_0}\partial_t R\,\psi_{i0}\tilde{I}_i + \frac{\varphi_l}{L_w}\partial_t(W\psi_1)\Bigg\}\,\mathrm{d}x \\
+\int_{x_c}^{L} & \Bigg\{\frac{1}{H_0}\left[\delta\psi_{ic}\tilde{I}_i - \frac{\delta h_c}{h_c}\psi_{ic}\tilde{G}_i\right]\left[-\partial_t h\varphi_{c^+} + h\partial_x\phi\partial_x\varphi_{c^+}\right] \\
&\quad +\delta\phi_{k'}\left[-\partial_t h\varphi_{k'} + h\partial_x\phi\partial_x\varphi_{k'}\right] \\
&\quad +(\delta h_c\varphi_{c^+} + \delta h_{k'}\varphi_{k'})\left[\partial_t\phi + \frac{1}{2}(\partial_x\phi)^2 + g(h-\check{H})\right]\Bigg\}\,\mathrm{d}x\,\mathrm{d}t = 0.
\end{aligned} \tag{4.47}$$

Arbitrariness of $\delta\psi_i$, $\delta\phi$ and δh leads to the following equations in shallow water:

$$\delta\phi_{k'}: \quad \partial_t h = -\partial_x(h\partial_x\phi), \qquad \forall x\in[x_c, L], \tag{4.48a}$$

$$\delta h_{k'}: \quad \partial_t\phi = -\frac{1}{2}(\partial_x\phi)^2 - g(h-H), \qquad \forall x\in[x_c, L]. \tag{4.48b}$$

On substitution of (4.48b) into (4.47), the last line vanishes. Moreover, the integral involving φ_{c^+} may be reduced to the integral for $x\in[x_c, x_c+\Delta x_{SW}]$ as $\varphi_{c^+}=0$ elsewhere. In that interval, Eq. (4.48a-b) are satisfied and may thus be substituted into (4.47) to yield

$$\begin{aligned}
&\int_0^T\int_{x_c}^{L}\frac{1}{H_0}\left[\delta\psi_{ic}\tilde{I}_i - \frac{\delta h_c}{h}\psi_{ic}\tilde{G}_i\right]\left[-\partial_t h\varphi_{c^+} + h\partial_x\phi\partial_x\varphi_{c^+}\right]\mathrm{d}x\,\mathrm{d}t \\
=&\int_0^T\int_{x_c}^{x_c+\Delta x_{SW}}\frac{1}{H_0}\left[\delta\psi_{ic}\tilde{I}_i - \frac{\delta h_c}{h}\psi_{ic}\tilde{G}_i\right]\left[\partial_x(h\partial_x\phi)\varphi_{c^+} + h\partial_x\phi\partial_x\varphi_{c^+}\right]\mathrm{d}x\,\mathrm{d}t \\
=&\int_0^T\frac{1}{H_0}\left[\delta\psi_{ic}\tilde{I}_i - \frac{\delta h_c}{h}\psi_{ic}\tilde{G}_i\right]\left[h\partial_x\phi\varphi_{c^+}\right]_{x_c}^{x_c+\Delta x_{SW}}\,\mathrm{d}t \\
=&\int_0^T -\frac{1}{H_0}\left[\delta\psi_{ic}\tilde{I}_i - \frac{\delta h_c}{h}\psi_{ic}\tilde{G}_i\right]\left[h\partial_x\phi\right]_{x=x_c}\,\mathrm{d}t,
\end{aligned} \tag{4.49}$$

as $\varphi_{c^+}(x_c + \Delta x_{SW}) = 0$, thus leading to boundary terms for the deep-water domain. The deep-water weak formulations with coupling boundary terms are thus given by

$$\begin{aligned}\int_0^{x_c} \delta_{i1}\frac{W}{L_w}\varphi_q\partial_t h \mathrm{d}x = \int_0^{x_c} \Bigg\{ \frac{1}{\Upsilon}\Bigg[& h\partial_x\varphi_q\partial_x\psi_j\tilde{M}_{ij} + \frac{1}{h}(\partial_x h)^2\psi_j\varphi_q\tilde{S}_{ij} \\ & - \partial_x h\left(\partial_x\varphi_q\tilde{D}_{ij}\psi_j + \varphi_q\partial_x\psi_j\tilde{D}_{ji}\right)\Bigg] \\ & + \Upsilon\left[\frac{1}{h}\varphi_q\tilde{A}_{ij}\psi_j - \frac{\delta_{i1}}{H_0}\varphi_q\frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x h\right]\Bigg\}\mathrm{d}x \\ + \frac{1}{H_0}\tilde{I}_i\Bigg\{ & \Big(h\partial_t R\Big)_{x=0} - \Big(hu\Big)_{x=x_c}\Bigg\},\end{aligned} \tag{4.50a}$$

and

$$\begin{aligned}\int_0^{x_c} \frac{\varphi_q}{L_w}\partial_t(W\psi_1)\mathrm{d}x = -\int_0^{x_c} \Bigg\{ \frac{1}{2\Upsilon}\Bigg[& \varphi_l\partial_x\psi_i\tilde{M}_{ij}\partial_x\psi_j - 2\partial_x\varphi_l\partial_x\psi_i\tilde{D}_{ij}\psi_j \\ & + \left(-\frac{\varphi_l}{h^2}(\partial_x h)^2 + \frac{2}{h}\partial_x h\partial_x\varphi_l\right)\psi_i\tilde{S}_{ij}\psi_j\Bigg] \\ & + \frac{\Upsilon}{H_0}\left[\varphi_l g\,(h-H) - \psi_1\frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x\varphi_l\right] \\ & - \frac{\Upsilon}{2h^2}\varphi_l\psi_i\tilde{A}_{ij}\psi_j\Bigg\}\mathrm{d}x \\ - \frac{1}{H_0}\Bigg\{ & \Big(\partial_t R\,\psi_i\tilde{I}_i\Big)_{x=0} + \left(\frac{hu}{h}\psi_i\tilde{G}_i\right)_{x=x_c}\Bigg\},\end{aligned} \tag{4.50b}$$

where we have substituted the shallow-water velocity $u = \partial_x\phi$. The coupling term is thus imposed weakly at the boundary $x = x_c$. Note that Eq. (4.50a) is a system of $n_z + 1$ equations, that is, one for each ψ_i. In particular, the case $i = 1$ describes the evolution of h while the cases $i > 1$ express the interior velocity potential $\psi_{i'}$ for $i' \in [2, n_z + 1]$ in terms of the surface velocity potential ψ_1 and depth h (*cf.* Eqs. (C.49) and (C.50) in Appendix C.3). The equations resulting from the weak formulations (4.50) are derived in Appendix C.3; they indeed are the same as the nonlinear potential-flow equations (4.11) and boundary condition (4.15) transformed with (4.19).

In order to obtain the coupling condition for shallow water, we distinguish the coupling node c from the other nodes in deep water by substituting into the variational principle (4.39)

$$\psi_i = \psi_{ic}\varphi_{c^-} + \psi_{iq'}\varphi_{q'}, \tag{4.51a}$$

$$h = h_c\varphi_{c^-} + h_{q'}\varphi_{q'}, \tag{4.51b}$$

with $q' \in [0, c^-[$. Substitution of the deep-water equations (C.51) into the terms involving φ_{c^-} in the resulting variational principle then yields

$$\begin{aligned}
0 = \int_0^T \int_0^L \delta\psi_{iq'} \Bigg\{ & \frac{1}{\Upsilon}\Bigg[h\partial_x\varphi_{q'}\partial_x\psi_j\tilde{M}_{ij} + \frac{1}{h}(\partial_x h)^2\psi_j\varphi_{q'}\tilde{S}_{ij} \\
& - \partial_x h\left(\partial_x\varphi_{q'}\tilde{D}_{ij}\psi_j + \varphi_{q'}\partial_x\psi_j\tilde{D}_{ji}\right)\Bigg] + \frac{\delta_{q'0}}{H_0}\varphi_0\partial_t R h_0\tilde{I}_i \\
& +\Upsilon\left[\frac{H_0}{h}\varphi_{q'}\psi_j\tilde{A}_{ij} - \frac{\delta_{i1}}{H_0}\varphi_{q'}\left[\partial_t h + \frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x h\right]\right]\Bigg\} \\
+\delta h_{q'}\Bigg\{ & \frac{1}{2\Upsilon}\Bigg[\varphi_{q'}\partial_x\psi_i\tilde{M}_{ij}\partial_x\psi_j + \left(-\frac{\varphi_{q'}}{h^2}(\partial_x h)^2 + \frac{2}{h}\partial_x h\partial_x\varphi_{q'}\right)\psi_i\tilde{S}_{ij}\psi_j \\
& - 2\partial_x\varphi_{q'}\partial_x\psi_i\tilde{D}_{ij}\psi_j\Bigg] - \frac{\Upsilon}{2h^2}\varphi_{q'}\psi_i\tilde{A}_{ij}\psi_j + \frac{\varphi_{q'}}{L_w}\partial_t(W\psi_1) \\
& +\frac{\Upsilon}{H_0}\left[\varphi_{q'}g\,(h-H) - \psi_1\frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x\varphi_{q'}\right] + \delta_{q'0}\frac{1}{H_0}\partial_t R\,\psi_{i0}\tilde{I}_i\Bigg\} \\
+\delta\phi_k & \Big[-\partial_t h\varphi_{k'} + h\partial_x\phi\partial_x\varphi_{k'}\Big] + \delta h_k\varphi_k\Big[\partial_t\phi + \frac{1}{2}(\partial_x\phi)^2 + g(h-\check{H})\Big] \\
+\frac{\delta\psi_{ic}}{H_0} & \int_0^{H_0}\left\{\tilde{\varphi}_i\left[\tilde{\varphi}_j h_c\partial_x\psi_{jc} - z\psi_{jc}\partial_x h_c\partial_z\tilde{\varphi}_j\right]_{x=x_c}\right\}\mathrm{d}z \\
+\frac{\delta h_c}{H_0} & \int_0^{H_0}\left\{z\psi_{ic}\partial_z\tilde{\varphi}_i\left[\frac{z}{h_c}\partial_x h_c\psi_{jc}\partial_z\tilde{\varphi}_j - \tilde{\varphi}_j\partial_x\psi_{jc}\right]_{x=x_c}\right\}\mathrm{d}z\,\mathrm{d}t.
\end{aligned} \tag{4.52}$$

The coupling condition (4.16) is transformed with (4.19) to lead to

$$\delta\psi_{ic}\tilde{\varphi}_i - \frac{\delta h_c}{h_c}z\partial_z\psi_c = \delta\phi_c, \tag{4.53}$$

which, once substituted into (4.52), leads to the shallow-water equations (4.12) with boundary condition

$$\delta\phi_c: \quad (hu)_{x=x_c} = \frac{1}{H_0}\left[h_c\partial_x\psi_{ic}\tilde{I}_i - \psi_{ic}\tilde{G}_i\partial_x h_c\right]_{x=x_c}. \tag{4.54}$$

The boundary condition (4.54) is imposed to the shallow-water domain as the flux at $x = x_c$ through

$$\begin{pmatrix} h \\ hu \end{pmatrix}_{x=x_c} = \begin{pmatrix} h_c \\ \dfrac{1}{H_0}\left[h_c\partial_x\psi_{ic}\tilde{I}_i - \psi_{ic}\tilde{G}_i\partial_x h_c\right] \end{pmatrix}. \tag{4.55}$$

The spatial discretisation now completed, we temporally discretise the variational principle and equations in the next section.

4.3.2 Temporal coupling strategies

We split the simulation time $[t^0, T]$, with t^0 and T the initial and final times respectively, into continuous time intervals $[t^n, t^{n+1}]$ of length Δt, with $t^n = t^0 + n\Delta t$. For a given initial solution $\left(\psi_1^0 = \psi_1(t^0),\ \psi_{i'}^0 = \psi_{i'}(t^0),\ h^0 = h(t^0),\ \check{\phi}^0 = \check{\phi}(t^0),\ \check{h}^0 = \check{h}(t^0)\right)$ the solutions are updated from time t^n to t^{n+1}, with $n \geq 0$, using the Symplectic-Euler scheme, which is shown to be a robust integrator for Hamiltonian systems [53]. For a Hamiltonian system with coordinate and momentum variables q and p respectively, the Symplectic-Euler scheme is defined as an implicit step for the first variable, and an explicit step for the second one, that is

$$q^{n+1} = q^n + \Delta t\frac{\partial H(q^{n+1}, p^n)}{\partial p^n} \tag{4.56a}$$

$$\text{and}\quad p^{n+1} = p^n - \Delta t\frac{\partial H(q^{n+1}, p^n)}{\partial q^{n+1}}. \tag{4.56b}$$

To apply the symplectic-Euler scheme (4.56) to the potential-flow and shallow-water equations, the space-discrete variational principle (4.39) is written in Hamiltonian form, as

$$0 = \delta\int_0^T \left[\boldsymbol{p}_1\frac{d\boldsymbol{h}}{dt} - H(\boldsymbol{h}, \boldsymbol{p}_i, W)\right] + \left[\check{\boldsymbol{p}}\frac{d\check{\boldsymbol{h}}}{dt} - \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})\right]\,\mathrm{d}t, \tag{4.57}$$

with

$$\boldsymbol{p}_i = [M_{lq}\psi_{iq}] \quad\text{and}\quad \boldsymbol{h} = [h_l] \qquad \text{for } l, q \in [0, c^-] \text{ and } i \in [1, n_z+1], \tag{4.58a}$$

$$\check{\boldsymbol{p}} = [M_{kr}\phi_r] \quad\text{and}\quad \check{\boldsymbol{h}} = [h_k] \qquad \text{for } k, r \in [c^+, N_l + c], \tag{4.58b}$$

and $H(\boldsymbol{h}, \boldsymbol{p}_i, W)$ and $\check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})$ the deep- and shallow-water Hamiltonians defined in Appendix C.4.

Setting

$$\boldsymbol{P} = \boldsymbol{p}_1 + \check{\boldsymbol{p}}, \tag{4.59a}$$

$$\boldsymbol{Q} = \boldsymbol{h} + \check{\boldsymbol{h}}, \tag{4.59b}$$

$$\bar{H}(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}_{i'}, W) = H(\boldsymbol{h}, \boldsymbol{p}_i, W) + \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}}), \tag{4.59c}$$

with $i' \in [2, n_z + 1]$, the variational principle (4.57) is equivalent to

$$0 = \delta \int_0^T \left[\boldsymbol{P} \frac{d\boldsymbol{Q}}{dt} - \bar{H}(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}_{i'}, W) \right] \mathrm{d}t = 0. \tag{4.60}$$

Variations with respect to the unknowns $\boldsymbol{P}$, $\boldsymbol{p}_{i'}$ and $\boldsymbol{Q}$ then lead to

$$\begin{aligned} \int_0^T \Bigg[& \delta\boldsymbol{P} \left(\frac{d\boldsymbol{Q}}{dt} - \frac{\partial \bar{H}(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}_{i'}, W)}{\partial \boldsymbol{P}} \right) - \delta \boldsymbol{p}_{i'} \left(\frac{\partial \bar{H}(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}_{i'}, W)}{\partial \boldsymbol{p}_{i'}} \right) \\ & -\delta \boldsymbol{Q} \left(\frac{d\boldsymbol{P}}{dt} + \frac{\partial \bar{H}(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{p}_{i'}, W)}{\partial \boldsymbol{Q}} \right) \Bigg] \mathrm{d}t, \end{aligned} \tag{4.61}$$

meaning that to apply the symplectic-Euler scheme (4.56), $\boldsymbol{Q}$ and $\boldsymbol{P}$ must be evaluated as follows:

$$\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^n + \Delta t \frac{\partial \bar{H}(\boldsymbol{P}^n, \boldsymbol{Q}^{n+1}, \boldsymbol{p}^*_{i'}, W^n)}{\partial \boldsymbol{P}^n}, \tag{4.62a}$$

$$\boldsymbol{P}^{n+1} = \boldsymbol{P}^n - \Delta t \frac{\partial \bar{H}(\boldsymbol{P}^n, \boldsymbol{Q}^{n+1}, \boldsymbol{p}^*_{i'}, W^n)}{\partial \boldsymbol{Q}^{n+1}}, \tag{4.62b}$$

$$0 = \frac{\partial \bar{H}(\boldsymbol{P}^n, \boldsymbol{Q}^{n+1}, \boldsymbol{p}^*_{i'}, W^n)}{\partial \boldsymbol{p}^*_{i'}}, \tag{4.62c}$$

where Eq. (4.62c) expresses $\boldsymbol{p}_{i'}$ in terms of $\boldsymbol{P}^n$ and $\boldsymbol{Q}^{n+1}$ (*cf.* Chap.3 for more details). Equations (4.62a-c) indicate that $\boldsymbol{h}$ and $\check{\boldsymbol{h}}$ must be evaluated implicitly while $\boldsymbol{p}_1$ and $\check{\boldsymbol{p}}$ must be evaluated explicitly. In order to apply the coupling conditions derived in section 4.3.1, $\delta\boldsymbol{P}$ and $\delta\boldsymbol{Q}$ are split as

$$\begin{aligned} 0 = \int_0^T \Bigg[& \delta\psi_{1q'} \left(M_{q'l} \frac{dh_l}{dt} - \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial (M^{-1}_{q'l} p_{1l})} \right) + \delta\psi_{1c} \left(M_{cl} \frac{dh_l}{dt} - \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial (M^{-1}_{cl} p_{1l})} \right) \\ & + \delta\phi_{k'} \left(M_{k's} \frac{dh_s}{dt} - \frac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial (M^{-1}_{k's} \check{p}_s)} \right) + \delta\phi_c \left(M_{ck} \frac{dh_k}{dt} - \frac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial (M^{-1}_{ck} \check{p}_k)} \right) \\ & - \delta\psi_{i'q'} \left(\frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial (M^{-1}_{q'l} p_{i'l})} \right) - \delta\psi_{i'c} \left(\frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial (M^{-1}_{cl} p_{i'l})} \right) \\ & - \delta h_{l'} \left(\frac{d\boldsymbol{p}_{1l'}}{dt} + \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial h_{l'}} \right) - \delta h_c \left(\frac{d\boldsymbol{p}_{1c}}{dt} + \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial h_c} \right) \\ & - \delta h_{k'} \left(\frac{d\check{p}_{k'}}{dt} + \frac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial \check{h}_{k'}} \right) - \delta \check{h}_c \left(\frac{d\check{p}_c}{dt} + \frac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial \check{h}_c} \right) \Bigg] \mathrm{d}t, \end{aligned} \tag{4.63}$$

with $l' \in [0, c^-[$ and $k' \in]c^+, \check{N}_x + c]$. In section 4.3.1, it was shown that

$$\begin{aligned}&\int_0^T \left[\delta\phi_c \left(M_{ck}\frac{d\check{h}_k}{dt} - \frac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial(M_{ck}^{-1}\check{p}_k)}\right)\right] \mathrm{d}t \\ &= \int_0^T \left[\delta\psi_{1c}\tilde{I}_1 + \delta\psi_{i'c}\tilde{I}_{i'} - \frac{\delta h_c}{h_c}\psi_{ic}\tilde{G}_i\right] (h_c\partial_x\phi_c)_{x=x_c} \mathrm{d}t,\end{aligned} \tag{4.64}$$

and

$$\begin{aligned}&\int_0^T \left[\delta\psi_{1c}\left(M_{cl}\frac{dh_l}{dt} - \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial(M_{cl}^{-1}p_{1l})}\right) - \delta\psi_{i'c}\left(\frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial(M_{cl}^{-1}p_{i'l})}\right)\right. \\ &\left. -\delta h_c\left(\frac{dp_{1c}}{dt} - \frac{\partial H(\boldsymbol{h}, \boldsymbol{p}_i, W)}{\partial h_c}\right)\right] \mathrm{d}t = \int_0^T \frac{\delta\phi_c}{H_0}\Big[h_c\partial_x\psi_i\tilde{I}_i - \psi_{ic}\tilde{G}_i\partial_x h\Big]_{x=x_c} \mathrm{d}t.\end{aligned} \tag{4.65}$$

Therefore the variational principle (4.61) leads to the following system of equations:

$$\begin{cases} M_{ql}\dfrac{dh_l}{dt} = \dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial(M_{ql}^{-1}p_{1l})} - \delta_{qc}\tilde{I}_1\,(hu)_{x=x_c}, & (4.66a) \\ M_{ks}\dfrac{d\check{h}_s}{dt} = \dfrac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial(M_{ks}^{-1}\check{p}_s)} - \delta_{kc}\dfrac{1}{H_0}\Big[h_c\partial_x\psi_i\tilde{I}_i - \psi_{ic}\tilde{G}_i\partial_x h\Big]_{x=x_c}, & (4.66b) \\ 0 = \dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial(M_{ql}^{-1}p_{i'l})} - \delta_{qc}\tilde{I}_{i'}\,(hu)_{x=x_c}, & (4.66c) \end{cases}$$

$$\begin{cases} \dfrac{dp_{1l}}{dt} = -\dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial h_l} - \delta_{lc}\left[\dfrac{\psi_i}{h}\tilde{G}_i(hu)\right]_{x=x_c}, & (4.66d) \\ \dfrac{dp_k}{dt} = -\dfrac{\partial \check{H}(\check{\boldsymbol{h}}, \check{\boldsymbol{p}})}{\partial \check{h}_k}. & (4.66e) \end{cases}$$

In section 4.2 and 4.3.1, we have shown that solving Eq. (4.66b) and (4.66e) for $\check{\boldsymbol{h}}$ and $\check{\boldsymbol{p}}$ is equivalent to solving Eq. (4.36) for $\check{h}$ and $\check{hu} = \check{h}\check{u}$ with boundary condition (4.55). Therefore, solving the system (4.66) is equivalent to solving

$$\begin{cases} M_{ql}\dfrac{dh_l}{dt} = \dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial(M_{ql}^{-1}p_{1l})} - \delta_{qc}\tilde{I}_1\left(\check{hu}\right)_{x=x_c}, & (4.67a) \\ \dfrac{d\check{h}_k}{dt} = -\dfrac{1}{\Delta x_{SW}}\left(F^{\check{h}}_{k+1/2}(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}}) - F^{\check{h}}_{k-1/2}(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}})\right) + \dfrac{1}{\Delta x_{SW}}S^{\check{h}}_k(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}}), & (4.67b) \\ 0 = \dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial(M_{ql}^{-1}p_{i'l})} - \delta_{qc}\tilde{I}_{i'}\left(\check{hu}\right)_{x=x_c}, & (4.67c) \end{cases}$$

$$\begin{cases} \dfrac{dp_{1l}}{dt} = -\dfrac{\partial H(\boldsymbol{h}, \boldsymbol{p}_1, \boldsymbol{p}_{i'}, W)}{\partial h_l} - \delta_{lc}\left[\dfrac{\check{hu}}{h}(\psi_1 \tilde{G}_1 + \psi_{i'}\tilde{G}_{i'})\right]_{x=x_c}, & (4.67d) \\ \dfrac{d\check{hu}_k}{dt} = -\dfrac{1}{\Delta x_{SW}}\left(F^{\check{hu}}_{k+1/2}(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}}) - F^{\check{hu}}_{k-1/2}(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}})\right) + \dfrac{1}{\Delta x_{SW}} S^{\check{hu}}_k(\check{\boldsymbol{h}}, \check{\boldsymbol{hu}}), & (4.67e) \end{cases}$$

with boundary conditions

$$\check{h}_{-1/2} = h_c, \tag{4.67f}$$

$$\check{hu}_{-1/2} = \frac{1}{H_0}\left[h_c \partial_x \psi_i \tilde{I}_i - \psi_{ic}\tilde{G}_i \partial_x h\right]_{x=x_c}, \tag{4.67g}$$

for the shallow-water equations (4.67b) and (4.67e).

Solving (4.67) with the symplectic-Euler scheme is equivalent to solving (4.62). Therefore, $\boldsymbol{h}$ and $\check{\boldsymbol{h}}$ must be evaluated at time t^{n+1}, while $\boldsymbol{p}_1$ and $\check{\boldsymbol{hu}}$ must be evaluated at time t^n. The explicit time dependency W being involved in $\boldsymbol{p}_1$, it is evaluated at time t^n (*cf.* Chap.3 for more details). The boundary term $(hu)_{x=x_c}$ in the deep-water weak formulations (4.67a-c-d) is the HLL flux

$$F^h_{-1/2}(h^{n+1}_{-1/2}, hu^n_{-1/2}) = \begin{cases} 0, & \text{if } S_L = S_R = 0, \\ (hu)^n_{-1/2^-}, & \text{if } S_L > 0, \\ (hu)^n_{-1/2^+}, & \text{if } S_R < 0, \\ \dfrac{S_R (hu)^n_{-1/2^-} - S_L (hu)^n_{-1/2^+} + S_L S_R (h^{n+1}_{-1/2^+} - h^{n+1}_{-1/2^-})}{S_R - S_L}, & \text{otherwise}, \end{cases} \tag{4.68}$$

with S_L and S_R the left and right speeds at the interface $x = x_c$ defined as

$$S_L = \min\left(\frac{hu^n_{-1/2^-}}{h^{n+1}_{-1/2^-}} - \sqrt{gh^{n+1}_{-1/2^-}}, \frac{hu^n_{-1/2^+}}{h^{n+1}_{-1/2^+}} + \sqrt{gh^{n+1}_{-1/2^+}}\right), \tag{4.69}$$

$$S_R = \max\left(\frac{hu^n_{-1/2^-}}{h^{n+1}_{-1/2^+}} + \sqrt{gh^{n+1}_{-1/2^-}}, \frac{hu^n_{-1/2^+}}{h^{n+1}_{-1/2^+}} + \sqrt{gh^{n+1}_{-1/2^+}}\right). \tag{4.70}$$

Therefore, $(hu)_{x=x_c}$ is implicit in $h^{n+1}_{-1/2\pm}$ and explicit in $hu^n_{-1/2\pm}$. We denote its semi-implicit time evaluation with the subscript $(hu)^*_{x=x_c}$. Similarly, the interior velocity described by $\boldsymbol{p}_{i'}$ depends on both $\boldsymbol{h}^{n+1}$ and $\boldsymbol{p}^n_1$, hence its semi-implicit evaluation $\boldsymbol{p}^*_{i'}$. Therefore, the coupled symplectic-Euler scheme reads

$$\begin{cases} M_{ql}^n h_l^{n+1} = M_{ql}^n h_l^n + \Delta t \dfrac{\partial H(\boldsymbol{h}^{n+1}, \boldsymbol{p}_1^n, \boldsymbol{p}_{i'}^*, W^n)}{\partial (M_{ql}^{n-1} p_{1l}^n)} - \delta_{qc} \tilde{I}_1 \left(\check{hu}^* \right)_{x=x_c}, & (4.71a) \\ 0 = \dfrac{\partial H(\boldsymbol{h}^{n+1}, \boldsymbol{p}_1^n, \boldsymbol{p}_{i'}^*, W^n)}{\partial (M_{ql}^{n-1} p_{i'l}^*)} - \delta_{qc} \tilde{I}_{i'} \left(\check{hu}^* \right)_{x=x_c}, & (4.71b) \\ h_{-1/2^-}^{n+1} = h_c^{n+1}, & (4.71c) \\ hu_{-1/2^-}^* = \dfrac{1}{H_0} \left[h_c^{n+1} \left(\partial_x \psi_1^n \tilde{I}_1 + \partial_x \psi_{i'}^* \tilde{I}_{i'} \right) - \left(\psi_{1c}^n \tilde{G}_1 + \psi_{i'c}^* \tilde{G}_{i'} \right) \partial_x h^{n+1} \right]_{x=x_c}, & (4.71d) \\ \check{h}_k^{n+1} = \check{h}_k^n - \dfrac{\Delta t}{\Delta x_{SW}} \Big(F_{k+1/2}^{\check{h}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) - F_{k-1/2}^{\check{h}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) & \\ \qquad\qquad - S_k^{\check{h}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) \Big), & (4.71e) \end{cases}$$

$$\begin{cases} p_{1l}^{n+1} = p_{1l}^n - \Delta t \dfrac{\partial H(\boldsymbol{h}^{n+1}, \boldsymbol{p}_1^n, \boldsymbol{p}_{i'}^*, W^n)}{\partial h_l^{n+1}} - \delta_{lc} \left[\dfrac{\check{hu}^*}{h^{n+1}} (\psi_1^n \tilde{G}_1 + \psi_{i'}^* \tilde{G}_{i'}) \right]_{x=x_c}, & (4.71f) \\ \check{hu}_k^{n+1} = \check{hu}_k^n - \dfrac{\Delta t}{\Delta x_{SW}} \Big(F_{k+1/2}^{\check{hu}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) - F_{k-1/2}^{\check{hu}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) & \\ \qquad\qquad - S_k^{\check{hu}} (\check{\boldsymbol{h}}^{n+1}, \check{\boldsymbol{hu}}^n) \Big), & (4.71g) \end{cases}$$

in which (4.71a-b-c-d-e) must be solved simultaneously, as well as (4.71f-g). The next section explains how the fully coupled system (4.71) is implemented.

4.4 Implementation of the fully-coupled system

The system (4.71) requires several iterations to deal with the implicit boundary terms, which is computationally costly. We choose to improve the computational cost by computing the shallow-water HLL flux $F_{k+1/2}^h$ explicitly, that is,

$$F_{k+1/2}^h = \begin{cases} 0, & \text{if } S_L = S_R = 0, \\ (hu)_{k+1/2^-}^n, & \text{if } S_L > 0, \\ (hu)_{k+1/2^+}^n, & \text{if } S_R < 0, \\ \dfrac{S_R (hu)_{k+1/2^-}^n - S_L (hu)_{k+1/2^+}^n + S_L S_R (h_{k+1/2^+}^n - h_{k+1/2^-}^n)}{S_R - S_L}, & \text{otherwise}, \end{cases} \quad (4.72)$$

with explicit left and right wave speeds

$$S_L = \min\left(\frac{hu^n_{k+1/2^-}}{h^n_{k+1/2^-}} - \sqrt{gh^n_{k+1/2^-}}, \frac{hu^n_{k+1/2^+}}{h^n_{k+1/2^+}} + \sqrt{gh^n_{k+1/2^+}}\right), \tag{4.73}$$

$$S_R = \max\left(\frac{hu^n_{k+1/2^-}}{h^n_{k+1/2^-}} + \sqrt{gh^n_{k+1/2^-}}, \frac{hu^n_{k+1/2^+}}{h^n_{k+1/2^+}} + \sqrt{gh^n_{k+1/2^+}}\right). \tag{4.74}$$

As a consequence, the boundary terms in (4.71a-b-f) and the fluxes in (4.71e) are explicit, thus simplifying the coupled system (4.71) to a fully explicit coupling in which the deep- and shallow-water equations may be solved separately through

$$\begin{cases} M^n_{ql}h^{n+1}_l = M^n_{ql}h^n_l + \Delta t\dfrac{\partial H(\boldsymbol{h}^{n+1},\boldsymbol{p}^n_1,\boldsymbol{p}^*_{i'},W^n)}{\partial(M^{n-1}_{ql}p^n_{1l})} - \delta_{qc}\tilde{I}_1\left(\check{hu}^n\right)_{x=x_c}, & (4.75a) \\ 0 = \dfrac{\partial H(\boldsymbol{h}^{n+1},\boldsymbol{p}^n_1,\boldsymbol{p}^*_{i'},W^n)}{\partial(M^{n-1}_{ql}p^*_{i'l})} - \delta_{qc}\tilde{I}_{i'}\left(\check{hu}^n\right)_{x=x_c}, & (4.75b) \end{cases}$$

$$p^{n+1}_{1l} = p^n_{1l} - \Delta t\frac{\partial H(\boldsymbol{h}^{n+1},\boldsymbol{p}^n_1,\boldsymbol{p}^*_{i'},W^n)}{\partial h^{n+1}_l} - \delta_{lc}\left[\frac{\check{hu}^n}{h^{n+1}}(\psi^n_1\tilde{G}_1 + \psi^*_{i'}\tilde{G}_{i'})\right]_{x=x_c}, \tag{4.75c}$$

$$h^{n+1}_{-1/2^-} = h^{n+1}_c, \tag{4.75d}$$

$$hu^*_{-1/2^-} = \frac{1}{H_0}\left[h^{n+1}_c\left(\partial_x\psi^n_1\tilde{I}_1 + \partial_x\psi^*_{i'}\tilde{I}_{i'}\right) - \left(\psi^n_{1c}\tilde{G}_1 + \psi^*_{i'c}\tilde{G}_{i'}\right)\partial_x h^{n+1}\right]_{x=x_c}, \tag{4.75e}$$

$$h^{n+1}_k = h^n_k - \frac{\Delta t}{\Delta x_{SW}}\Big((hu)^n_{k+1/2} - (hu)^n_{k-1/2}\Big), \tag{4.75f}$$

$$\begin{aligned} \check{hu}^{n+1}_k = \check{hu}^n_k - \frac{\Delta t}{\Delta x_{SW}}\Big(&\left[\frac{(hu)^n}{h^{n+1}} + \frac{1}{2}g(h^{n+1})^2\right]_{k+1/2} - \left[\frac{(hu)^n}{h^{n+1}} + \frac{1}{2}g(h^{n+1})^2\right]_{k-1/2} \\ &- \frac{1}{2}g\left((h^{n+1}_{k+1/2})^2 - (h^{n+1}_{k-1/2})^2\right)\Big), \end{aligned} \tag{4.75g}$$

$$(hu)^{n+1})_{x=x_c} = (\check{hu}^{n+1})_{-1/2}. \tag{4.75h}$$

Equations (4.75a-b) must be solved simultaneously to update the deep-water depth and interior velocity implicitly. This is done using Firedrake, in which we solve the semi-discrete weak-formulations, that is, continuous x–integrals and discrete z– and time integrals (*cf.* Chapter 3 for more details). Equation (4.75c) is also solved with Firedrake semi-implicitly to update the deep-water surface velocity potential. Equations (4.75d-e) update the boundary condition for the shallow-water flux at $x = x_c$. Equation (4.75f) updates the shallow-water depth fully explicitly. Finally, hu is updated semi-explicitly in shallow water with Eq. (4.75g) and the flux at $x = x_c = x_{-1/2}$ is saved as the next boundary condition for deep water. This scheme is slightly less accurate but faster than the symplectic-Euler scheme (4.71), and still more accurate than a fully explicit scheme such as the forward-Euler scheme. Results are discussed in the next section.

4.5 Results

In Chapter 3, the nonlinear potential-flow equations were solved with the symplectic-Euler scheme in the case of a domain closed by a vertical wall on the right-hand side. The results showed good agreement with experimental data. In addition, the semi-symplectic-Euler scheme for nonlinear shallow-water equations as implemented in (4.75) is verified against an exact solution in Appendix C.5. In this section, we analyse the effectiveness with which the nonlinear coupling reduces the wave reflection through energy absorption on the shallow-water beach. The effect of the beach is first observed by comparing coupled simulations to deep-water simulations with a vertical wall at the coupling position $x = x_c$. The dimensions in each case, as well as the wavemaker and topography properties, are given in Table 4.1.

	Length $[m]$		Topography				wavemaker		
	x_c	L	H_0 $[m]$	x_B $[m]$	s_B	$H(x_c)$ $[m]$	γ $[m]$	T $[s]$	L_w $[m]$
Coupling	11.0	14.0	1.0	3.0	0.1	0.2	0.02	1.13	1.0
Wall	/	11	1.0	3.0	0.1	0.2	0.02	1.13	1.0

Table 4.1: Characteristics of the coupled and wall domains, including topography and wavemaker settings.

With a period $T = 1.1339s$, the wavelength of the waves based on a linear dispersion analysis for a constant rest depth H_0 is $\lambda \approx 2.0m$. We set the deep-water resolution to $\Delta x_{DW} = \lambda/40 = 0.05m$ and the shallow-water resolution to $\Delta x_{SW} = (\Delta x_{DW})^2 = 0.0025m$. Time runs from $t = 0$ to $t = 110T$ with a time step of $\Delta t = 0.001s$, thus satisfying the stability condition given in Chapter 3:

$$\Delta t \leq \frac{2}{\omega}. \tag{4.76}$$

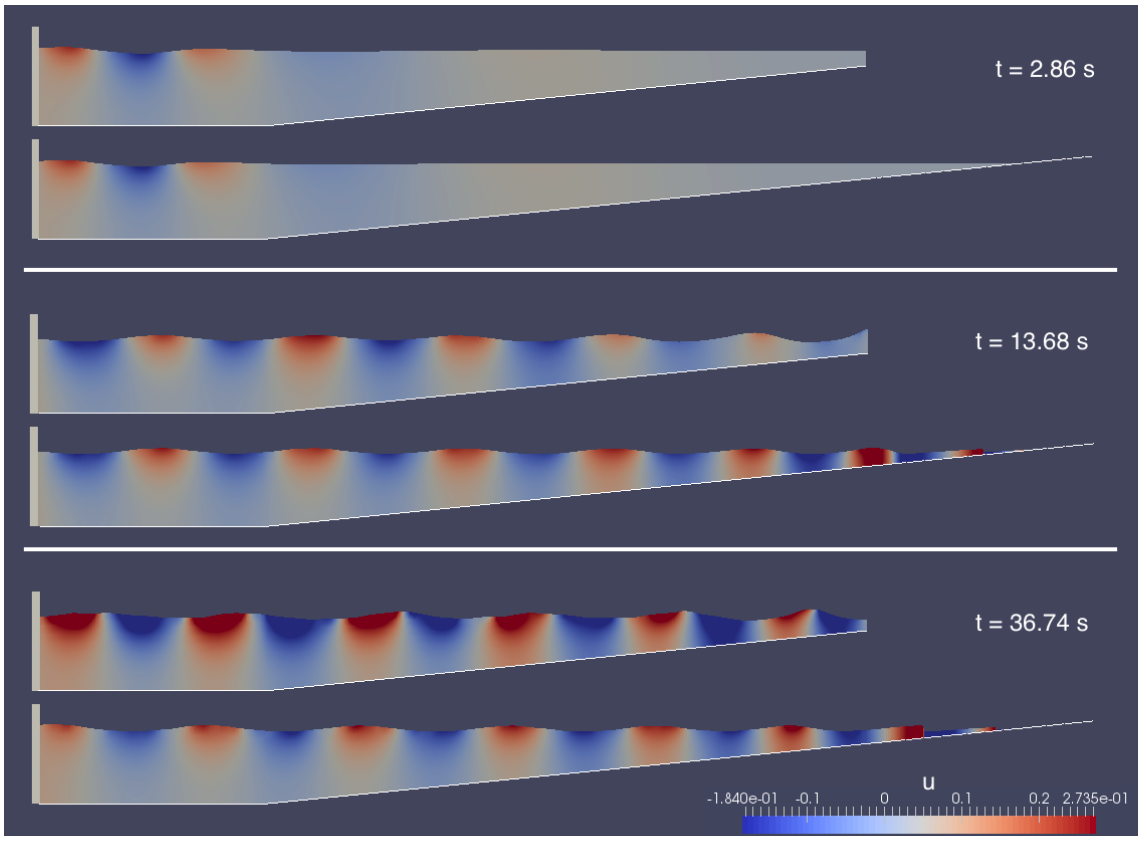


Figure 4.4: Comparison of the wave propagation in a domain closed by a vertical wall and a domain with wave absorption on the beach at different times.

Figure 4.4 compares the waves propagation in the domain with a vertical wall and in the coupled domain. In the top sub-figure, at time $t = 2.86$s, the waves are neither disturbed by the wall nor the beach as they have not yet reached the boundary. Hence, the solutions are the same in the two models. However, when the waves have travelled past x_c at, *e.g.*, 13.6s (see middle subfigure), the waves of the first model have reflected against the wall while those in the coupled model have entered the shallow-water domain. As time proceeds, the effect of wall reflection increasingly affects the deep-water waves of the first model, while no obvious reflection is noticeable in the coupled model due to wave breaking on the beach (*e.g.* at time $t = 36.74$s). The objective of avoiding disturbance of the deep-water domain by reflected waves has thus been achieved thanks to the coupling to the shallow-water beach.

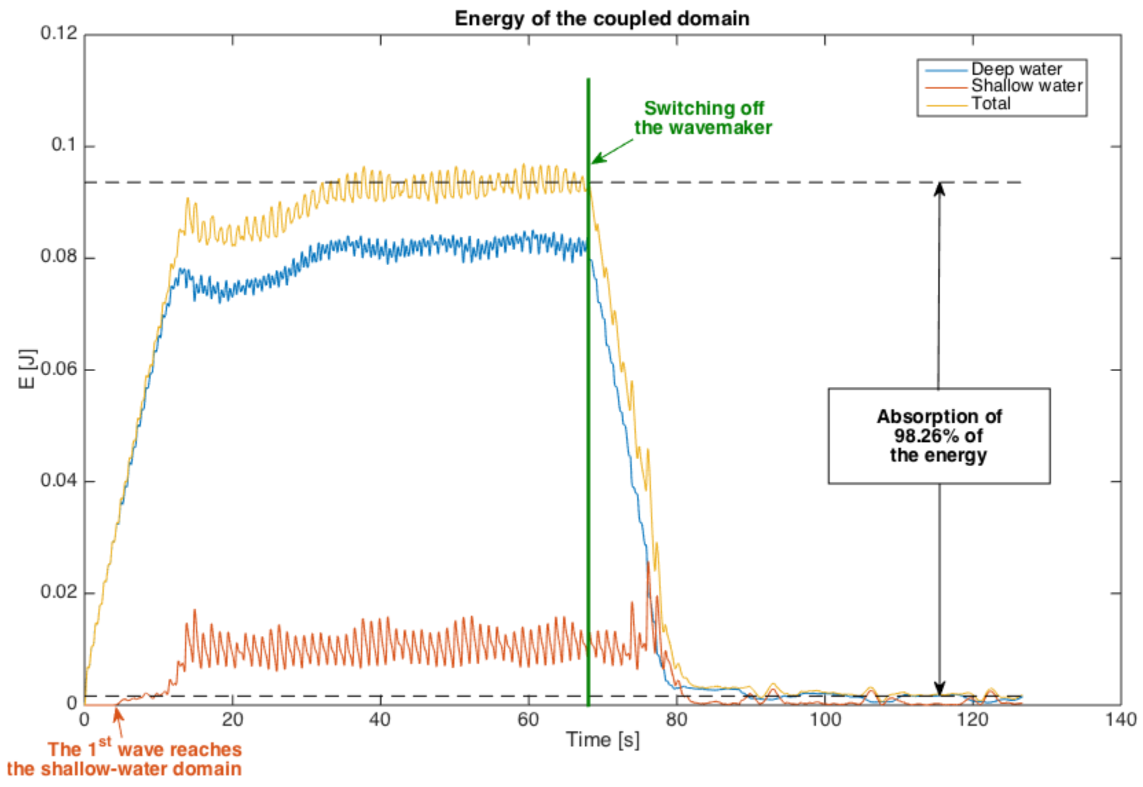


Figure 4.5: Energy of the deep- and shallow-water models. At time $t = 0.0$s, the system is at rest. The wavemaker is turned on at $t > 0.0$s, leading to an increase of energy, first in deep water, followed by an increase of energy in shallow water. The absorption of energy by the beach leads to a conservation of the global energy. At $t = 68.03$s, the wavemaker is turned off, resulting in an energy decrease that results from wave breaking on the beach.

In order to quantify the efficiency of the beach in absorbing the waves, the energies of the deep- and shallow-water domains in the case of the coupled model are displayed in Fig. 4.5. At time $t = 0.0$s, the tank is at rest and the energy is null. Immediately after switching on the wavemaker, energy is added to the numerical domain due to the wavemaker motion. As the waves first travel through the deep-water domain, as shown in Fig. 4.4, energy is first imparted to only the deep-water domain. The moment when the first wave reaches the shallow-water domain is characterised by a clear increase of energy in the shallow-water domain, as indicated by the orange arrow in Fig. 4.5. As a consequence, the gradient of the deep-water energy decreases, confirming that some deep-water energy is transferred into the shallow-water domain through the coupling interface. The time t_{sw} taken for the first wave to reach the shallow-water domain may be computed from the fastest wave velocity as

$$t_{sw} \approx x_c/\sqrt{gH_0} = 11/\sqrt{9.81 \times 1.0} = 3.5\text{s}, \tag{4.77}$$

where we have used the shallow-water limit to compute the velocity, as the longest waves travel fastest. This estimation of t_{sw} is consistent with Fig. 4.5. For $t > t_{sw}$, the energy continues to increase in the deep-water domain, meaning that the energy given by the wavemaker is still larger than the one absorbed by the beach. However, when higher amplitude waves reach the shallow-water domain, an equilibrium is reached between the energy given by the wavemaker and the one absorbed by the beach, leading to a net and approximate conservation of the total energy. This is explained by the fact that, in deep water, waves are generated by the wavemaker while similar waves are travelling through the coupling interface, hence resulting in a net balance between the addition and absorption of energy. In shallow water, the waves arriving from the coupling interface are absorbed by the beach when breaking, hence the energy oscillates around a constant value. One can however notice some long-period oscillations in the total energy, which are due to the long-wave reflection. In Chapter 5, Fourier spectra are used to analyse these long-wave reflections. To confirm that most of the energy is absorbed by the beach, the wavemaker is switched off after 60 wavemaker periods, *ie.* at $t = 68.03$s (*cf.* green arrow in Fig. 4.5). As expected, the energy in the deep-water domain immediately decreases with the same gradient as its initial energy increase. Therefore, the coupling interface behaves as a transparent boundary for the deep-water domain, confirming that the coupling is consistent. However, in the shallow-water domain, the energy does not decrease immediately because the last waves travelling from the deep-water domain still bring in energy. After $t = 6.24$s, which is the time required for the last wave to travel from the wavemaker to the coupling interface, an increase of energy in the shallow-water domain, resulting from the last wave entering the domain, can be seen. It takes $t \approx 1.5$s for this wave to travel to the beach before being absorbed, thereafter resulting in a decrease of the shallow-water energy. Finally, Fig. 4.5 shows that the total energy does not decrease to its initial value of zero, but to approximately $E = 1.6mJ$. This is only 1.7% of the total energy before switching off the wavemaker, meaning that the beach absorbs about 98.3% of the energy in that case, while

100% of the energy was conserved in the case of a rigid wall in Chapter 3. The efficacy of the beach to absorb wave energy is further explored in the next chapter and compared with validation against experiments.

4.6 Conclusions and discussions

The numerical tank presented in this chapter is the first-ever reported fully nonlinear coupling between deep-water (potential-flow) and shallow-water equations. The variational approach ensures stable simulations of waves travelling through the coupling interface Γ_c in a smooth way, thus simulating a real sea state. The energy variations show that the coupling interface behaves as a transparent boundary for the deep-water waves while the beach absorbs more than 98% of the energy. The model may therefore be used as a cost-improved alternative of the deep-water model presented in Chapter 3, since it reduces the numerical domain and thereby saves substantial computational resources. The choice of absorbing waves with a topographical beach makes the numerical tank similar to the experimental tank at MARIN, where waves are generated by a piston wavemaker and absorbed through wave breaking on the beach. Wave generation can thus first be tested in the numerical wave tank before being used in the experimental tanks, thereby ensuring that the wavemaker motion will by design generate the waves in the target area.

Some improvements can, however, increase the efficiency of the present numerical tank. First, the semi-symplectic Euler scheme used for optimal computational cost may be extended to the symplectic-Euler scheme or a higher-order scheme by conserving the implicit evaluation of the HLL flux in shallow water, as explained in section 4.4 [54]. For instance, the second-order Störmer-Verlet scheme implemented in Chapter 3 in deep water may be applied to the coupled model by following the method described in section 4.3.2. Second, the implementation of the shallow-water solver is not yet optimised and could be improved with parallel programming. To facilitate the parallelisation of the shallow-water solver, a detailed tutorial of the code is given in Chapter 6. Note that the deep-water solver may be compiled in parallel without any additional modification, as it is solved within Firedrake. Third, as explained in section 4.2, the coupling location is chosen *a priori* from the wavelengths, which has the advantage of being fixed and prescribed but is not optimal in the case of irregular waves. Indeed, to make sure that the vertical variations of the velocity potential are negligible at the coupling point for irregular waves, dynamic determination of x_c can be made depending on the vertical structure of the velocity potential in the potential-flow domain, thus ensuring that the shallow-water assumption holds at the coupling point. Another solution was proposed by Cotter and Bokhove [33] and extended by Gagarina *et al.* [55], who proposed a new model that encompasses both the three-dimensional potential-flow water-water model and the depth-averaged shallow-water model as limiting systems. Consequently, the coupling point sets itself dynamically; both Cotter and Bokhove [33] and Gagarina *et al.* [55] also derive hydraulic-jump conditions for this new model, which conditions equal the usual shallow-water ones in the shallow-water limit. Future work includes extension of this method to the nonlinear coupling presented herein. An extension of the present model to a 3D tank is also part of future work. Since the 3D deep-water model is already implemented in Chapter 3 and the surface shallow-water model is a well-known problem (see for example [25, 102]), only the extension of the coupling interface remains as future work; the same method as in 2D can be used and will result in extra terms involving the y–derivatives of the solutions. Finally, the coupled model will be compared with and validated against experiments to study its efficacy in absorbing various types of waves, including irregular, short and steep waves. For that purpose, and for validation of future extended models, various test cases were performed in the experimental tank of Delft University of Technology (DUT). Description of these experiments and validation of the present numerical tank is deferred to Chapter 5.

Chapter 5

Experimental validation of the numerical wave tank

5.1 Introduction

The numerical models derived in Chapter 3 and Chapter 4 successfully showed that extreme waves can not only be generated with our (numerical) modelling approach but also absorbed at the shallow-water beach. In this chapter, the limits of the previously presented models are further explored with a view towards making future improvements.

The objective of the maritime industry is to build safer ships and platforms using the simulation of realistic sea states, in which waves have irregular frequency and amplitude. As highlighted in the conclusions of Chapter 4, one limit of our coupling process is that the coupling interface must be set *a priori* and stays static during the simulations. Indeed, the coupling interface must be set relatively to the wavelengths and amplitudes to ensure that waves do not break in the deep-water domain and that the velocity of the waves is depth-uniform before entering the shallow-water domain. The coupling process might therefore not be accurate in a sea state with varying wave frequency and amplitude, due to an inadequate location of the coupling interface. In this chapter, the numerical simulation and absorption of irregular waves are validated against experiments in the case of three different irregular wave profiles.

In addition, the ability of the numerical beach to absorb and partially reflect long and short waves is further investigated. When an ocean wave reaches a real (sand) beach, part of it is reflected back into the ocean; the reflection factor, which is the ratio between absorbed and reflected waves, depends on the beach slope and on the wavelengths: damping of short waves is more efficient than damping of long waves. We thus provide experimental data of long- and short-wave absorption/reflection from a beach with a plausible gradient of 10% and compare the experimental reflection to the one obtained with our numerical wave tank.

The comparison of the numerical simulations of short waves with the experimental measurements will also help the maritime industry to predict amplitude modulation resulting from the translational motion of the piston wavemaker. As explained in the conclusions of Chapter 3, deep-water waves, *i.e.,* short waves relative to the depth, have a non-uniform vertical-velocity profile and thus cannot be generated properly from a piston wavemaker with uniform height-independent horizontal motion. This is a common issue in experimental wave tanks, as short waves do not behave as intended when generated by a piston. By comparing the numerical and experimental data for short waves, we aim to evaluate the ability of the numerical tank to reproduce the disturbance of the waves due to the piston motion of the wavemaker, with a view towards using the model as a prediction tool to determine *a priori* whether the waves are long enough to be accurately generated in the experimental tanks of the maritime industry.

Finally, approaching the breaking-wave limit in the deep-water model is of interest to the maritime industry to test impact of steeper, thus stronger, waves on marine structures. When approaching the breaking-wave limit in the potential-flow model derived in Chapter 3, the overturning crest of the waves cannot be captured by our continuous finite-element method, thereby leading to numerical instabilities. In order to estimate the steepness limitation of the coupled model, waves of various amplitudes are generated and compared to measurements. These measurements of waves approaching the breaking-wave limit will also be available for validation of future breaking-wave models.

To study the accuracy of our numerical wave tank for the four aforementioned challenges, we have conducted experiments in the wave tank of the Delf University of Technology (TUD). The experimental set-up is described in Section 5.2. Several measuring tools are installed in the tank to record the experiments; their utilisation and precision are described in section 5.3. To study the accuracy of the numerical wave tank to generate and absorb various types of waves, several wave profiles are generated in the wave tank, and specified in Section 5.4. The corresponding experimental and numerical wavemaker inputs are computed in Section 5.5. The ability of the numerical tank to capture and absorb irregular, long, short and steep waves is studied by comparison with the experimental measurements in Section 5.6. Finally, the conclusions of the above four analyses are summarised in Section 5.7.

5.2 Experimental set-up

To analyse the performance of the coupled numerical tank, a slanted beach is placed in the wave basin of TUD, in which waves are generated from a piston wavemaker; that is, a structure moving with translational motion in the x–direction. As shown in Fig. 5.1, an absorbing wall is installed behind the wavemaker to absorb the back wave. The total basin is 85m-long, 2.75m-wide and filled with 1.0m water depth.

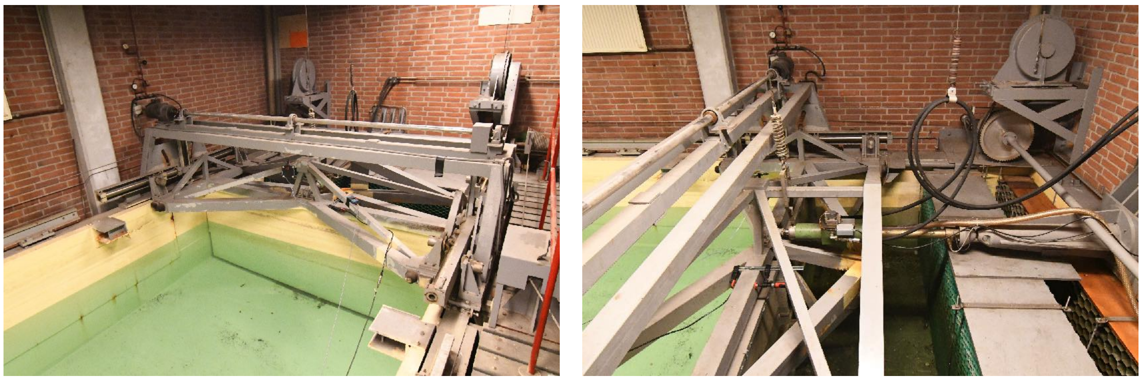


Figure 5.1: Piston wavemaker (left) and absorbing wall (right) in the tank of TUD.

The wooden beach is installed at location $x_B \approx$ 20m from the wavemaker, with a constant slope of 0.1, as shown in Fig. 5.2. With $x_B \approx$ 20m, the total length of the domain at rest is $L_x \approx$ 30m, which is short enough to minimize the phase error resulting from the simulations (as shown in Chapter 3) while leaving enough space between the wavemaker and the beach to allow the addition of a structure, such as a wind turbine or a ship, in the deep-water area. Probes are placed adjacent to various free-surface locations to measure the wave height: the choice of their positions is explained in section 5.3.1.

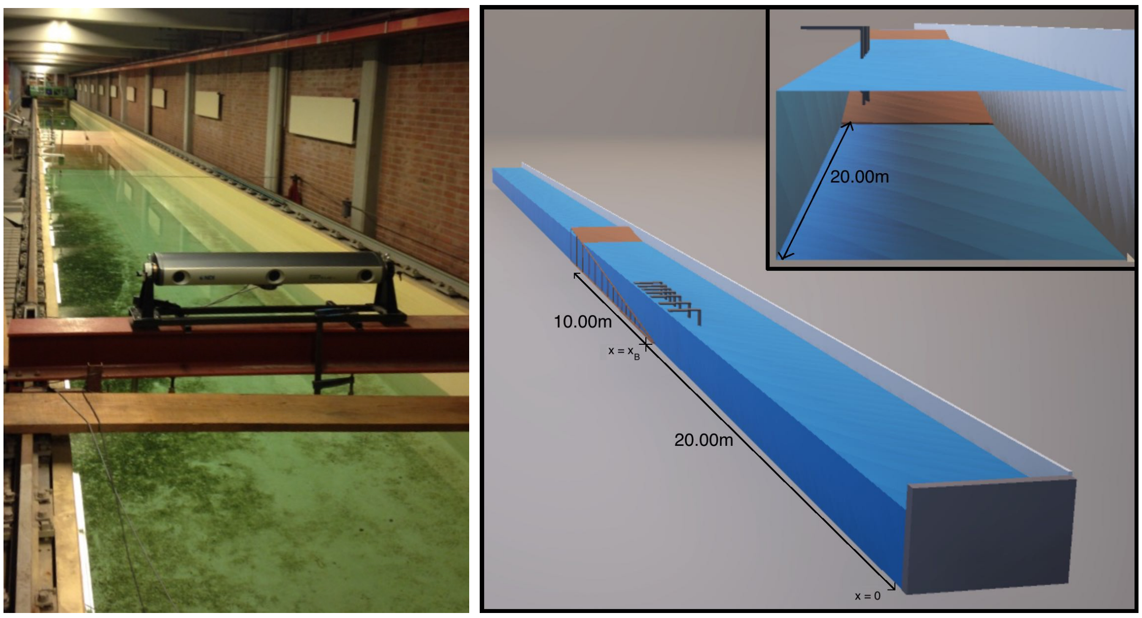


Figure 5.2: Experimental tank of TUD (left) and its schematic representation (right) with custom-made beach and L–shape probes (grey).

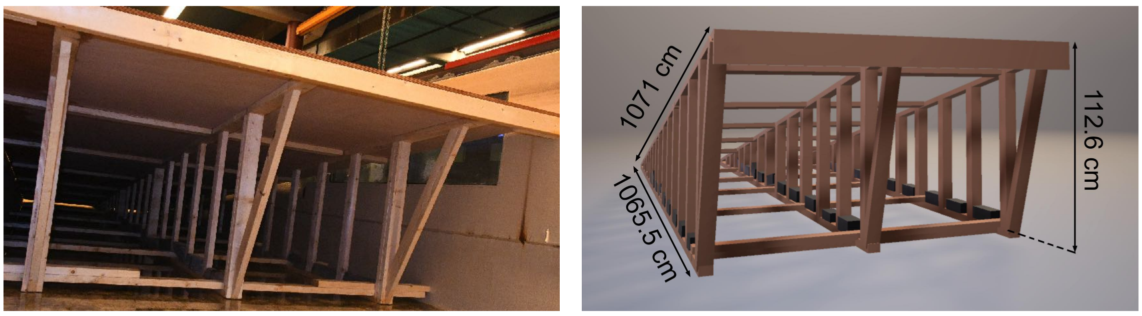


Figure 5.3: Photo (left) and schematic (right) of the experimental beach and the supporting frame which is weighted with lead blocks (black) to preclude unwanted bed motion.

The 1071cm beach is built using seven 153cm-wide boards, leaving enough space for the waterline to travel along the beach. The dimensions of the beach are given in Fig. 5.3. The real slope s_p given by these dimensions is

$$s_p = \frac{1}{3}\left(\frac{112.6}{1065.5} + \frac{112.6}{\sqrt{1071^2 - 112.6^2}} + \frac{\sqrt{1071^2 - 1065.5^2}}{1065.5}\right) = 0.104, \qquad (5.5)$$

which we have approximated by $s_p = 0.1$. The exact position of the beach is also estimated from laser measurements as $x_B = 20.24$m.

To avoid beach motion resulting from the impact of waves, the structure is pinned down against the bottom of the tank by a total of 500kg of lead weights. As shown in Fig. 5.3, these weights are regularly located on the beach structures; additional 90kg of blocks are placed intermittently along the line of the tank-beach intersection, where the height of the beach is too low to place the weights below it. An analysis is carried out in section 5.3.2 to ensure that the extra weights do not disturb the propagation of waves.

Finally, a gap of 0.02m is allowed between the tank bottom and the start of the beach (*cf.* Fig 5.4) to ensure that there is no difference in water level on both sides of the beach at $x = L_x$, as this would result in additional forces on the beach structure.

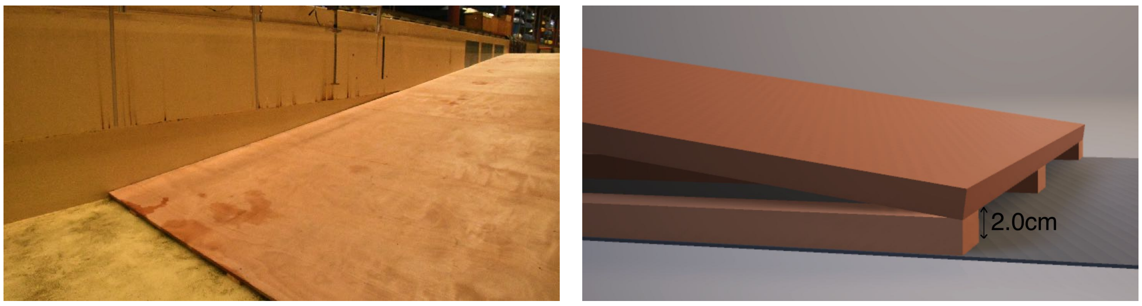


Figure 5.4: Left: Start of the beach. A gap of 0.02m between the tank bottom and the start of the beach ensures equal water level on both sides of the beach. Right: dimensions of the start of the beach. The water depth at rest at $x \leq x_B$ is 1.00m.

In order to admit reproduction of the wave profiles in the numerical model, the experimental wavemaker motion and wave elevation are recorded. These measurements are detailed in the next section.

5.3 Collection of experimental data

Several measuring instruments were installed to record the wavemaker and fluid motions in order to reproduce and compare the experimental results to the numerical simulations of the coupled tank. In this section, we present the instruments and estimate their precision.

5.3.1 Measuring equipment

Wavemaker accelerometer

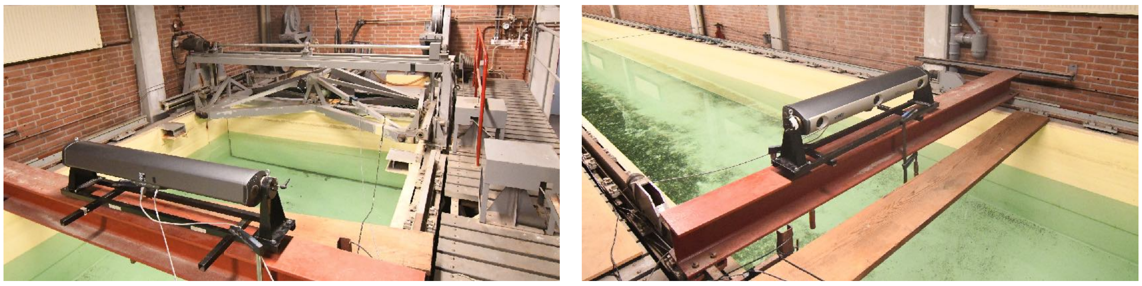


Figure 5.5: Accelerometer facing the piston wavemaker to measure its motion and acceleration.

Measuring both the wavemaker motion and acceleration ensures that the input signal is the intended one. Moreover, the numerical model requires the wavemaker motion and velocity as inputs of the simulations. Recording the wavemaker behaviour is thus necessary to reproduce the experimental waves in the numerical wave tank. To this end, an accelerometer is installed in front of the wavemaker to optically measure its position at a frequency of 100Hz (*cf.* Fig.5.5). The measured motion and acceleration are used to compute the numerical motion and velocity of the wavemaker in Section 5.5.2 in order to compare the calculated numerical waves to the corresponding experimental waves.

Wave probes

To measure the free-surface elevation, wave probes are placed at various locations in the wave tank (see Fig. 5.6). Figure 5.7 shows the two types of probes available for measuring the wave elevation. They are set at approximately 0.3m from the tank side wall, and record the free-surface elevation at a frequency of 100Hz with an accuracy of order $O(1\text{mm})$. Three probes are placed in the deep-water area ($x < x_B$), three above the beach ($x > x_B$) and one at the start of the beach ($x = x_B$). Figure 5.6 shows the location of the probes in the experimental tank. Their targeted locations are $x_1 = 15.0$m, $x_2 = 17.0$m, $x_3 = 19.0$m, $x_4 = 20.0$m, $x_5 = 21.0$m, $x_6 = 22.0$m and $x_7 = 23.0$m. However, the locations of the probes need to be known as precisely as possible (*i.e* at a precision of at least order $O(\Delta x_{DW})$) to ensure a fair comparison between the experiments and the numerics. To minimize the error, repeated measurements of their location are conducted with a laser and detailed in Table 5.1. Measurements 1 and 2 are made from the wavemaker to the probes, while measurement 3 is made from probe 1 to the other probes to limit the error made on long-distance measurements. The averaged measurements are used to set the probe locations in the numerical model.

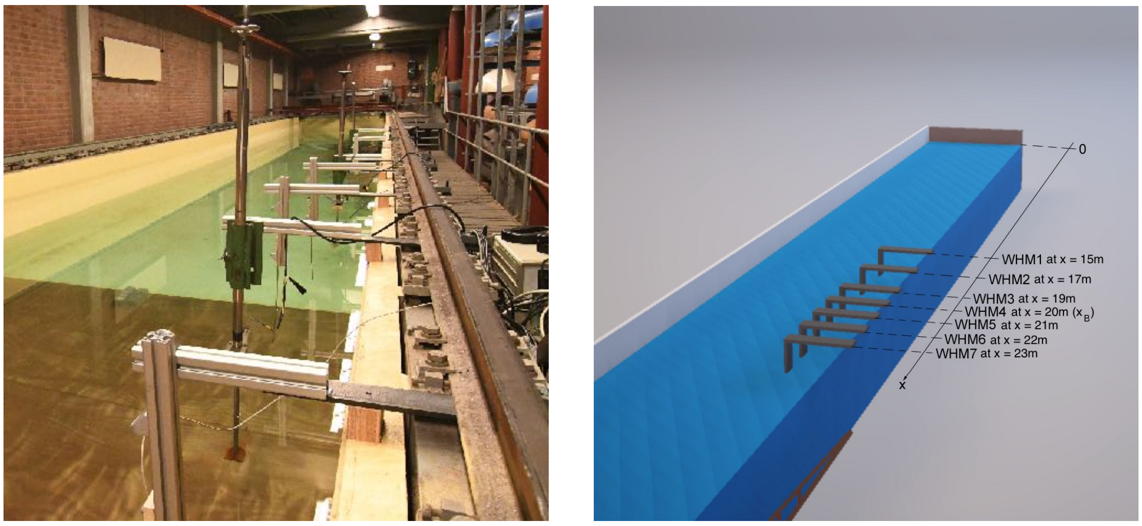


Figure 5.6: Wave probes in the experimental tank (left) and schematic representation of their location (right).

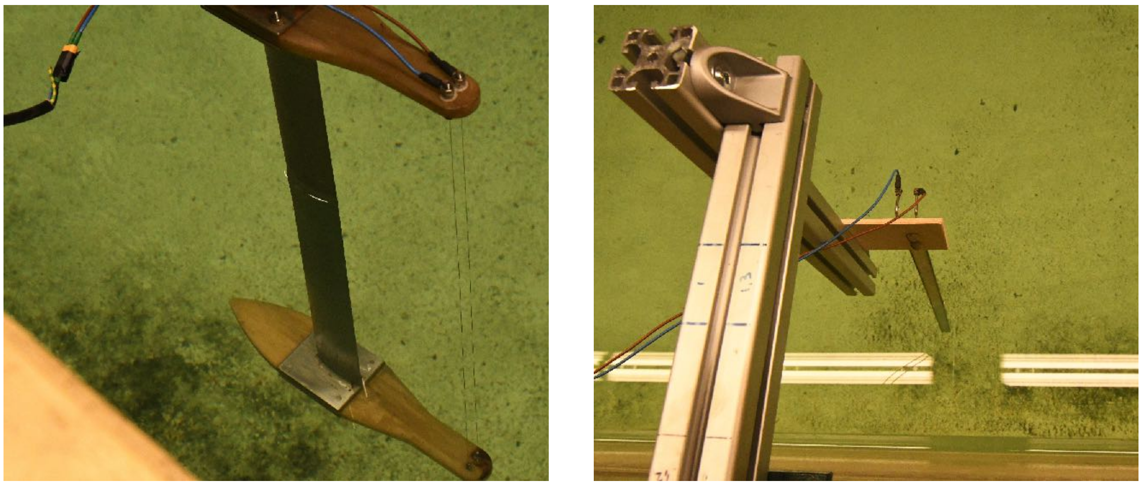


Figure 5.7: Photographs of the probes used for measuring the free-surface elevation. Left: type used for WHM1, WHM3 and WHM6. Right: type used for WHM2, WHM4, WHM5 and WHM7.

	measurement 1 [m]	measurement 2 [m]	measurement 3 [m]	average [m]
x_1 (WHM1)	15.000	14.998	/	14.999
x_2 (WHM2)	17.082	17.077	17.090	17.083
x_3 (WHM3)	19.038	19.037	19.040	19.038
x_4 (WHM4)	20.015	20.017	20.010	20.014
x_5 (WHM5)	21.075	21.079	21.086	21.080
x_6 (WHM6)	22.025	22.020	22.022	22.022
x_7 (WHM7)	23.154	23.154	23.161	23.156

Table 5.1: Laser measurements of the probe locations.

Table 5.2 shows that the deviation of each measurement from the averaged value is of order $O(10^{-3})$m. As the shortest wavelengths is of about 1.0m, the error coming from the probe location precision is negligible.

	deviation 1 [m]	deviation 2 [m]	deviation 3 [m]
x_1 (WHM1)	0.001	0.001	/
x_2 (WHM2)	0.001	0.006	0.007
x_3 (WHM3)	0.002	0.003	0.0010
x_4 (WHM4)	0.000	0.003	0.004
x_5 (WHM5)	0.005	0.001	0.006
x_6 (WHM6)	0.001	0.004	0.002
x_7 (WHM7)	0.003	0.003	0.004
Average	0.002	0.003	0.004

Table 5.2: Precision of the probe-location measurements.

Video capture of the waterline

To measure the location of the waterline, we place a gauge on the beach, as shown in Fig. 5.8. The gauge is 0.05m high and a correction must therefore be applied to the measured position to estimate the real waterline location.

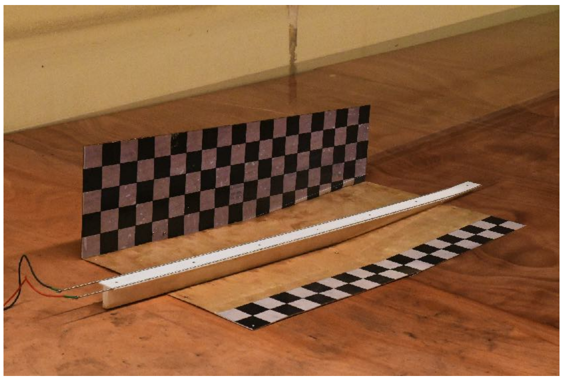


Figure 5.8: Beach probe to estimate the location of the waterline.

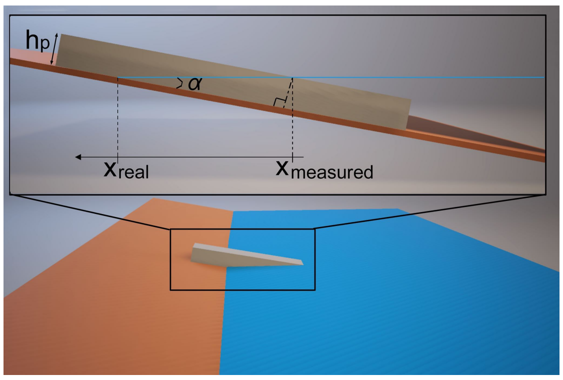


Figure 5.9: Estimation of the real waterline position from the measured location.

To increase the precision of the waterline estimation, an additional high-speed video of the waterline evolution is recorded at high resolution 1920×1080 pixels at 60 frames per second. In order to estimate the location of the waterline from the video, a chequered board is placed on, and perpendicular to, the beach, as shown in Fig. 5.10. Each square is 5cm $\times 5$cm, and the exact location of the chequered board is known (*cf.* Fig. 5.10). Therefore, the distance Δx_{WL} between the end of the beach and the waterline may be estimated from the chequered board.

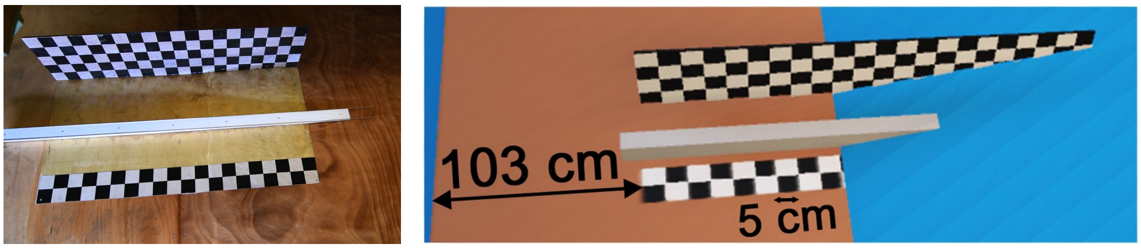


Figure 5.10: Chequered board (left) and its location relative to the beach (right).

As shown in Fig. 5.11, the coordinate x_{WL} of the waterline may then be computed through

$$x_{WL} = x_B + (1.071\text{m} - \Delta x_{WL})s_p. \tag{5.6}$$

The combination of the two aforementioned measurement methods enables estimation of the waterline position.

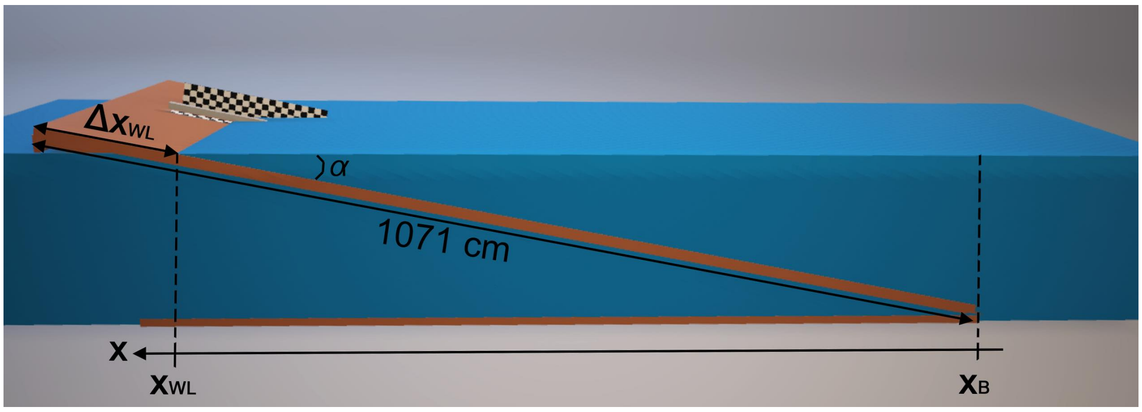


Figure 5.11: Estimation of the waterline coordinate from the chequered board.

5.3.2 Measuring uncertainty

The measuring precision and the differences between the experimental and numerical set-up must be taken into consideration to fairly analyse any discrepancy between experimental and numerical results.

The first difference between the experimental and numerical models is the gap of 0.02m below the start of the beach (*cf.* Fig. 5.4) in the experimental tank, while the numerical beach starts at $z = 0$m. The vertical resolution in the numerical tank will thus be set larger than 0.02m at the start of the beach.

A second difference concerns the additional weights pinning down the baseline of the beach, which change the water height locally. The weights are placed as in the left subfigure of Fig. 5.12, that is, on both sides and at the middle of the beach (configuration 1); this configuration is the one used in the experiments. However, since one of the weights is close to the probe WHM4, a test is made to estimate its effect on the wave behaviour. To this end, repeated measurements are made in this configuration and compared with the configuration 2 pictured in the right subfigure of Fig. 5.12, where two weights are placed at the middle of the beach width and one on the side opposite to the probe.

Figure 5.13 shows that the difference between the two configurations is not more significant than between repeated tests of the same configuration. In both cases, the difference is only a few millimetres, which is expected from the probe-measurement precision of order $O(1\text{mm})$. Therefore, we can assume that the weights do not significantly disturb the wave behaviour.

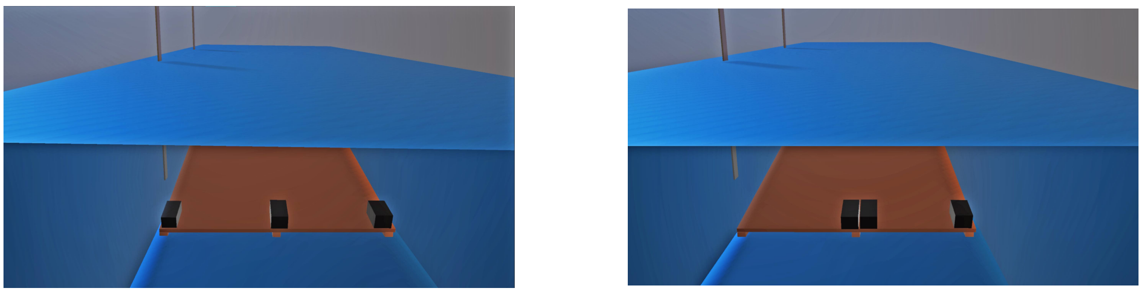


Figure 5.12: Left: Configuration 1: weights are placed on both sides and in the middle of the beach to hold it in place. Configuration 2: two weights are placed in the middle of the beach and one on the side opposite to the probes.

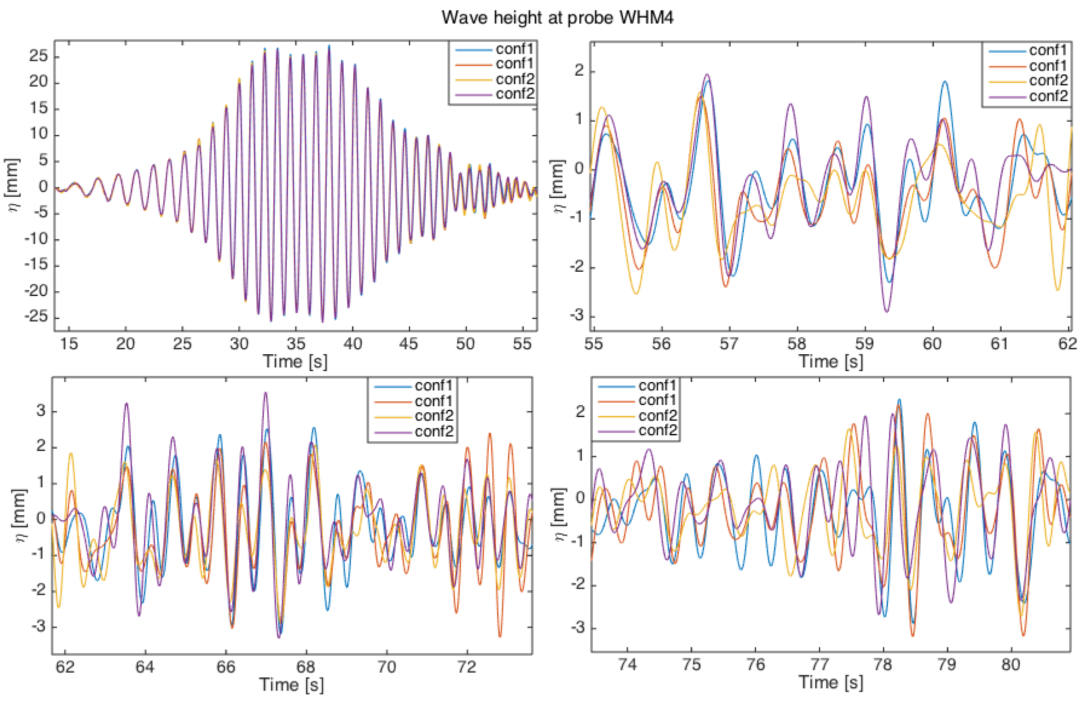


Figure 5.13: Wave-height measurements at probe WHM4 with configuration 1 (left of Fig. 5.12) (blue and orange curves for repeated measurements) and configuration 2 (right of Fig. 5.12) (yellow and violet curves for repeated measurements) in various measurement-time intervals.

Finally, the measurement error induced by the accuracy of the probes is minimized by both the repetition of the measurements and a calibration carried out regularly, as explained in section 5.3.3.

5.3.3 Calibration of the wave gauges

A calibration is carried out before the measurements are made to ensure that the measured signal corresponds to the actual water height. Probes 1, 3 and 6 can be moved up and down manually, to change the position of their intersection with the waterline, with a precision of order $O(0.001\text{m})$. To be calibrated, their position is set to the lowest height (-0.180m) in calm water (h = 1.00m). Then, every 30s, their position is changed manually, with a step of 0.020m.

The measured heights are compared to the expected ones for each of the three probes, and the sensitivity s_k at each vertical location z_k is then computed using

$$s_k = \frac{z_{k\,\text{measured}}}{z_{k\,\text{expected}}}, \tag{5.7}$$

and the corresponding accuracy is obtained through:

$$a_k = \frac{s_k}{\overline{s_k}} - 1, \tag{5.8}$$

where $\overline{s_k}$ is the averaged sensitivity over the probe height. Due to corrosion of the two wires (*cf.* Fig. 5.7), the sensitivity of the probe is not constant along its height. Therefore, the error in the measurements of the trough of the wave will vary slightly from the one at the crest.

To increase the accuracy of the probe measurements, the signal obtained from the amplifier may be adjusted through a calibration factor. This calibration factor is first set to 1.0, and then multiplied by the averaged sensitivity $\overline{s_k}$ along the probe height. At the next calibration, the updated calibration factor is obtained by multiplying the previous calibration factor with the updated averaged sensitivity. The probes are assumed to be accurately calibrated when the sensitivity tends to 1, with constant accuracy over the probe height; that is, when the calibration factor stays constant from one calibration to the next.

The other four probes (WHM2, WHM4, WHM5 and WHM7) cannot be moved vertically as easily as probes WH1, WH3 and WH6. In order to calibrate them, we generate a wave with estimated frequency and height, and we compare the signal measured by probes WHM2, WHM4, WHM5 and WHM7 not only to the expected height but also to the ones measured by probes WHM1, WHM3 and WHM6 (already calibrated). Again, the ratio between the measured height and the expected one enables adjustment of the calibration factor in order to increase the accuracy of the measurements.

Once acceptable accuracy is obtained, the calibration is repeated only once per experimental session, to ensure that the probes still capture the water height accurately. As the accuracy of the probes is of the order $O(1\text{mm})$, the measurement error for the low-amplitude waves (*i.e.,* $H_s \leq 0.1\text{m}$) is more significant than for higher-amplitude waves. To limit this error, the tests are conducted at least twice and compared.

5.4 Wave specification

The wave profiles used as input of the wave generator are defined through the wavelength λ and height H_s of the waves. Through these two parameters, the wave frequency ω, period T, velocity c and steepness s may be estimated by employing linear theory to be:

$$\omega = \sqrt{\frac{2\pi g}{\lambda}\tanh\left[\frac{2\pi H_0}{\lambda}\right]}, \qquad T = \frac{2\pi}{\omega}, \qquad c = \frac{\lambda\omega}{2\pi}, \qquad s = \frac{H_s}{\lambda}. \tag{5.9}$$

The evolution of the free-surface deviation η from the mean depth H_0 at the wavemaker is then obtained through a sinusoidal profile as

$$\eta(R,t) = \frac{H_s}{2}\sin(\omega t). \tag{5.10}$$

To better observe the reflection from the beach, we stop the wavemaker motion when the first generated wave is expected to reflect at the beach surface. As shown in Fig. 5.14, the length of the basin at rest (time t_0) is $L_x = 30$m with WHM1 placed at $x_1 = 15$m. If the last wave is generated when the first wave reflects at the beach (at time t_1), then there is a distance of 30m between the first and the last waves. Therefore, the reflection of the first wave meets the last generated wave at WHM1 (time t_2). Then, the first wave is reflected at the wavemaker boundary while the last wave reflects at the beach, and they meet again at WHM1 at t_3. Subsequently, only reflected waves are measured at the probes.

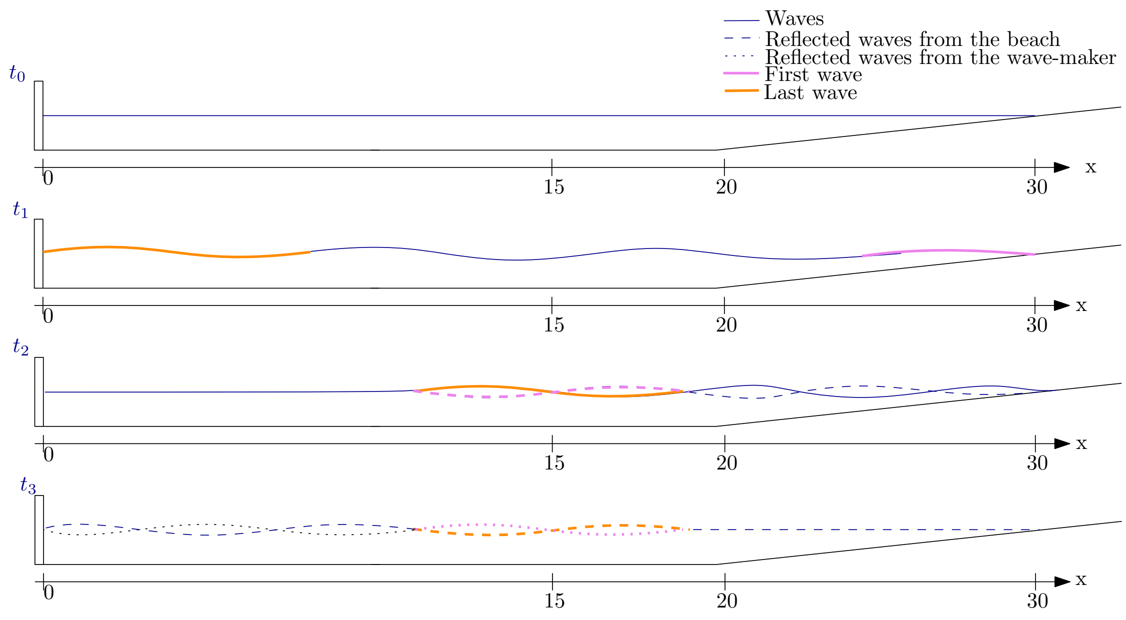

Waves
Reflected waves from the beach
Reflected waves from the wave-maker
First wave
Last wave
t_0
t_1
t_2
t_3
0
15
20
30
x

Figure 5.14: Configuration of the number of waves to be generated. At time t_0, water is at rest; at time t_1, the first wave reaches the beach and the wavemaker is turned off; at time t_2, the last generated wave meets the reflection of the first wave at the probe WHM1; and, at time t_3, the reflection of the last wave meets the first wave again at probe WHM1.

The maximum number N of waves to generate in order to satisfy this configuration may be computed from the wavelength of the waves as follows:

$$N = \frac{L_x}{\lambda}. \tag{5.11}$$

Various configurations, both for regular and irregular waves, are tested to approach the breaking- and deep-water limits and answer the objectives described in the introduction. References of the regular and irregular wave specifications are given in Tables 5.3 and 5.4 respectively. The values of Table 5.3 are estimated from the linear wave theory, using (5.9); the characteristics of the waves in the basin will vary from the expected ones. To obtain these regular and irregular waves in the wave tank, the corresponding wavemaker-input signal must be computed. This is done by defining a *transfer function* between the wavemaker stroke and the wave height, as explained in the next section.

Regular waves

Test case	Wavelength λ [m]	Frequency ω [rad/s]	Period T_p [s]	Velocity c_p [m/s]	Wave height H_s [m]	Steepness [$-$]
1.1.1	4.00	3.76	1.67	2.39	0.050	0.0125
1.1.2					0.100	0.0250
1.1.3					0.200	0.0500
1.2.1	2.00	5.54	1.13	1.76	0.050	0.0250
1.2.2					0.070	0.0350
1.2.3					0.100	0.0500
1.3.1	1.00	7.85	0.80	1.25	0.030	0.030
1.3.2					0.050	0.050

Table 5.3: Test cases for regular waves. Three wavelengths are considered (second column). The corresponding frequencies, periods, and velocities are estimated with Eq. (5.9). For each wavelength, several characteristic wave heights are tested (sixth column), yielding varying steepness (seventh column), including one approaching the breaking-wave limit ($s_w = 0.05$).

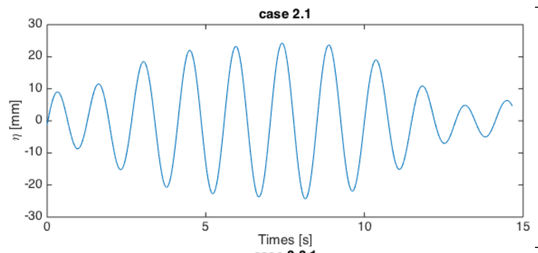
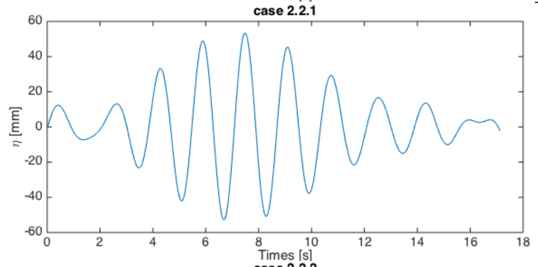
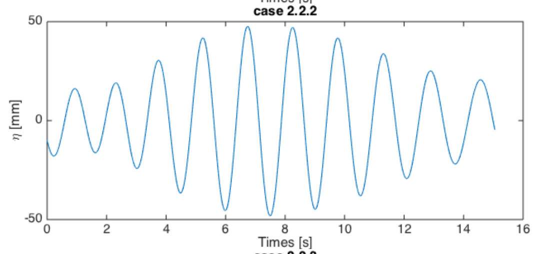
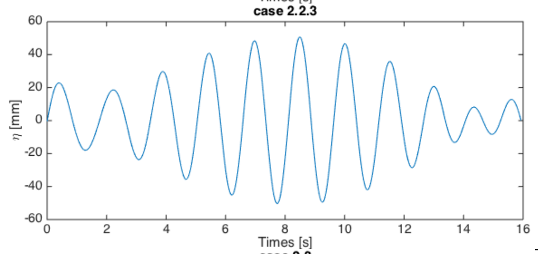
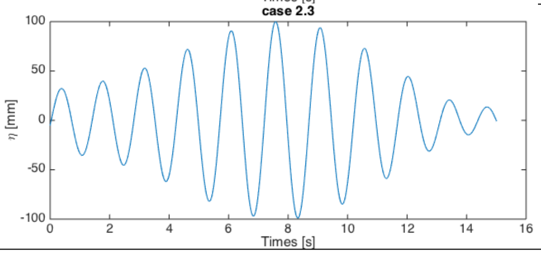
Test case		H_s [m]	T_p [s]	Wave profile
2.1		0.05	1.56	
2.2	2.2.1	0.1	1.56	
	2.2.2			
	2.2.3			
2.3		0.2	1.56	

Table 5.4: Test cases for irregular waves with random frequency and amplitude.

5.5 Wavemaker input

5.5.1 Experimental wavemaker input

In order to generate the expected waves in the basin, the wavemaker must be calibrated by computing the transfer function; that is, the ratio between the wavemaker stroke and the resulting wave height in the basin. The method used to estimate the transfer function is detailed in section 5.5.1. Once the required wavemaker stroke is known, a filter is applied at the start and at the end of the wavemaker signal to smooth out its motion and hence to reduce the generation of additional frequencies. This is done by applying a ramp function to the signal, as explained in section 5.5.1.

Determination of the transfer function

To calibrate the wavemaker, various prescribed input signals, expressed in volts, are sent to the signal generator. The resulting waves are measured at two probes (WHM2 and WHM6). As the waves are not dampened much between x_2 and x_6, these two probes, previously calibrated, should measure similar wave heights. By fitting the measured signals with a sinusoidal function, the wave amplitude and period are extracted from the records. Table 5.5 shows the measured periods and wave amplitudes at WHM2 (x_2) and WHM6 (x_6), as well as the measured (U_{meas}) input signals from the wavemaker, in response to various input signals U_{sent}.

T	U_{sent}	U_{meas}	$h(x_2)$	$h(x_6)$
[s]	[V]	[V]	[mm]	[mm]
1.13	1.1	1.09	60.66	62.1
1.67	1.5	1.48	63.7	61.6
2.0	0.50	0.49	16.20	15.99
3.0	3.0	2.96	61.8	60.35

Table 5.5: Prescribed (U_sent) and measured (U_meas) wavemaker inputs, measured wave period T and amplitudes $h(x_2)$ at probe WHM2 and $h(x_6)$ at probe WHM6.

The first thing to note from Table 5.5 is that the prescribed and measured wavemaker signals are the same when measured to a precision of order $O(10^{-2}\text{V})$. The wave height varies a few millimetres between the two probes, which is consistent with the results from the calibration in section 5.3.3. The transfer function RAO, in Volts/mm, is obtained from Table 5.5 by computing the ratio between the measured wavemaker stroke and the wave amplitude for each wave period:

$$RAO_{2,6} = \frac{U_{meas}}{h(x_{2,6})}. \tag{5.12}$$

Table 5.6 and Fig. 5.15 give the RAO computed from the measurements at WHM2 and WHM6 (RAO$_2$ and RAO$_6$ respectively) for each wave period of Table 5.5.

T [s]	RAO$_2$ [V/mm]	RAO$_6$ [V/mm]
1.13	0.0180	0.0176
1.67	0.0232	0.0240
2.0	0.0302	0.0306
3.0	0.0479	0.0490

Table 5.6: Transfer function (RAO) computed at WHM2 and WMH6 for various wave periods.

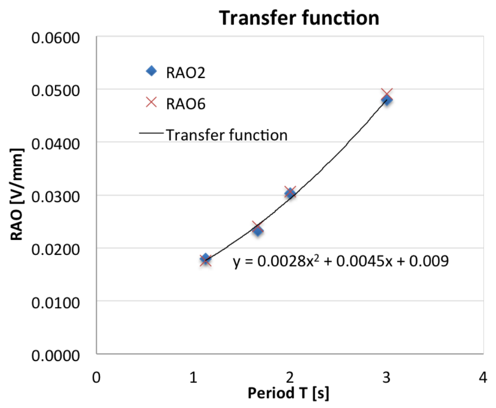


Figure 5.15: Transfer function (RAO) as a function of the wave periods, obtained by interpolating the values of Table 5.6 with a 2nd–order polynomial.

The trend line in Fig. 5.15 is a polynomial of order 2 describing the RAO [piston[V]/amplitude[mm]] as a function of T [s] :

$$\text{RAO} = 0.0028T^2 + 0.0045T + 0.009 \quad \text{(V/mm)}. \tag{5.13}$$

The input of the wavemaker required to obtain a specific wave height in the basin may be computed using Eq. 5.13:

$$S_{piston} = \frac{H_s}{2}\left[0.0028T^2 + 0.0045T + 0.009\right] \quad \text{(V)}. \tag{5.14}$$

Before generating the waves specified in section 5.4, Eq. 5.14 is verified. We aim to generate a wave with period $T = 1.13$s and wave height $H_s = 70$mm. Using Eq. 5.14, the stroke of the wavemaker should be 0.62V. The measured wave height at probes WHM2 and WHM6 are displayed in Table 5.7.

T [s]	Expected H_s [mm]	WM input [V]	Measured WM [V]	Amplitude $h(x_2)$ [mm]	Amplitude $h(x_6)$ [mm]
1.13	70	0.62	0.61	35.78	35.77

Table 5.7: Test of the transfer function.

From Table 5.7, the probe WHM2 measures a wave height of $2 \times 35.78 = 71.56$mm while WHM6 measures $2 \times 35.77 = 71.54$mm. The precision of the probes being of order $O(1\text{mm})$, these measurements confirm that the transfer function given in Eq. 5.14 leads to the expected wave height and may be used to translate the wave specified in section 5.4 into the corresponding wavemaker stroke. Note that a small variation between the expected and measured wave height would not be an issue as only the measured signals will be used and compared to the numerics. Therefore, the transfer function is required only to approximate the intended wave profiles. Before sending the wavemaker signal to the wave generator, the start and the end of the wavemaker motion must be smoothed. This is done next.

Determination of the ramp function

The wavemaker must be switched on and off smoothly to reduce the generation of extra frequencies in the wave spectrum. Several methods, such as tanh-function smoothing, may be used to gradually turn on and off the wavemaker signal. The custom at the wave tank of the TUD is to start the wavemaker with two identical but out-of-phase waves so that the initial signal is actually null. By shifting progressively the phase shift, the two waves slowly add until they reach the intended amplitude. We applied this method both at the start and at the end of our wavemaker signal.

To this end, a ramp time T_{up} during which the wavemaker is turned on or off is set. In our case, it is set to twice the wave period, that is,

$$T_{up} = 2T. \tag{5.15}$$

Then, the two input signals are defined as follows:

$$U_1(t) \quad = \quad \frac{U}{2}\sin(\frac{2\pi}{T}t) \qquad \text{and} \qquad U_2(t) \quad = \quad \frac{U}{2}\sin(\frac{2\pi}{T}t + \epsilon(t)\pi), \tag{5.16}$$

where U is the wavemaker stroke obtained with the transfer function Eq. 5.14 and ϵ is the time-dependent phase-shift between the two waves. By definition, the sum of U_1 and U_2 is the intended wavemaker input when the waves are in phase (that is, when $\epsilon = 0$) and is zero when U_1 and U_2 are out of phase (that is, when $\epsilon = 1$).

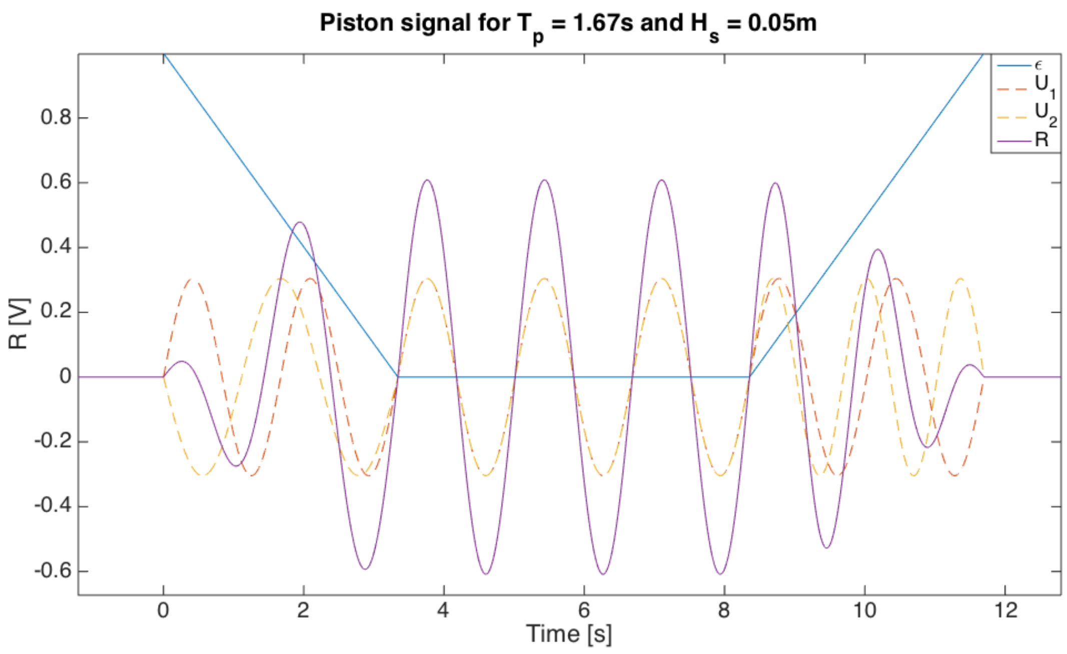


Figure 5.16: Wavemaker-input signal built from U_1 and U_2 with the ramp function for the case $T_p = 1.67$s, $H_s = 0.05$m.

For a smooth start and end of the wavemaker signal, ϵ is defined as

$$\epsilon(t) = \begin{cases} 1 - \dfrac{t}{T_{up}}, & \text{if } t \leq T_{up}, \\ 0, & \text{if } T_{up} \leq t \leq T_{stop} - T_{up}, \\ \dfrac{t - T_{stop} - T_{up}}{T_{up}}, & \text{if } t \geq T_{stop} - T_{up}, \end{cases} \tag{5.17}$$

where T_{stop} is the time at which the wavemaker is intended to stop. Figure 5.16 shows the shift function $\epsilon(t)$, together with the two building signals $U_1(t)$ and $U_2(t)$ and the resulting wavemaker input $R(t)$ in the case 1.1.1, that is, for $T_p = 1.67$s and $H_s = 0.05$m (*cf.* Table 5.3). Since the period of the regular waves is $T = 1.67$s, the ramp time is $T_{up} = 3.34$s. The final wavemaker input also includes a zero signal before and after the wavemaker motion in order to allow time both to start the record (before the wavemaker motion) and to let the waves propagate (after the last wave generation) to measure the reflection from the beach, as explained in section 5.4. An example of the full wavemaker input to generate waves of peak period $T_p = 1.67$s and characteristic wave height $H_s = 0.05$m is given in Fig. 5.17.

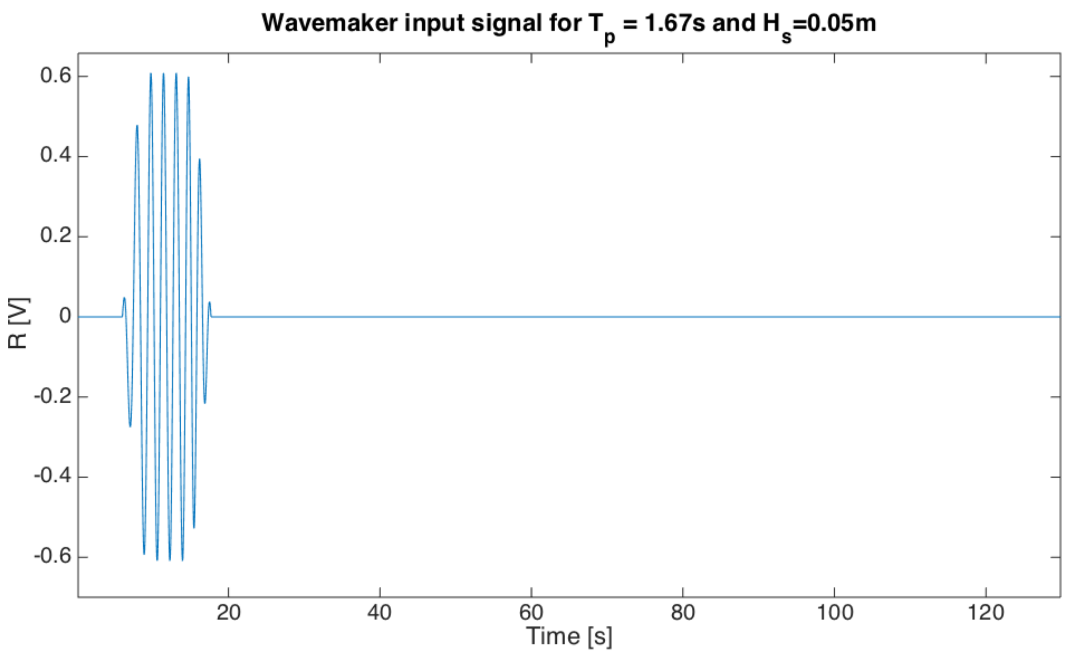


Figure 5.17: Full wavemaker-input signal to generate waves with $T_p = 1.67$s and $H_s = 0.05$m.

5.5.2 Numerical wavemaker input: extraction of the measured wavemaker motion and velocity

In order to validate the simulations against the experiments, the measured wavemaker motion and calculated velocity are used as input of the model. First, the measured wavemaker motion, in millimetres, is filtered with a low-pass filter with cut-off frequency of 30Hz to remove the measurement noise. To save some computational time, the first 3.0s of the filtered signal, during which the wavemaker is static, are then ignored.

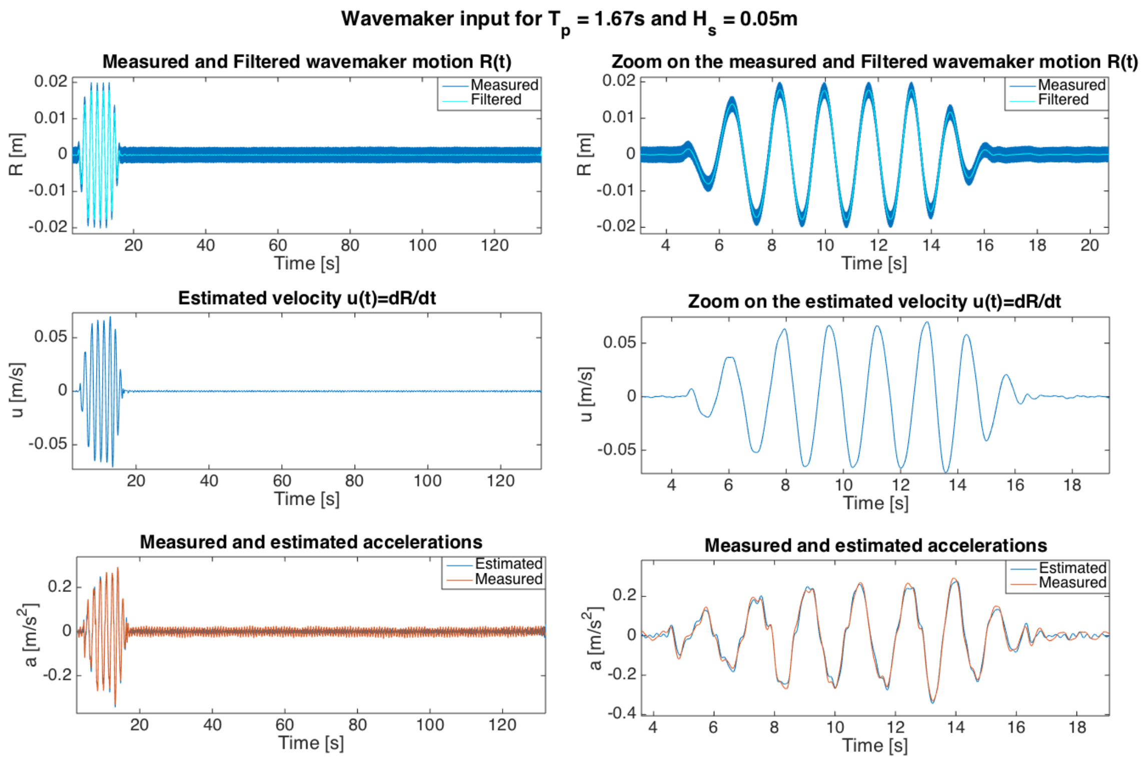


Figure 5.18: Example of measured and estimated wavemaker motion (top row), velocity (middle row) and acceleration (bottom row) used as input of the numerical model.

Figure 5.18 (top row) shows the truncated measured wavemaker motion together with the truncated filtered signal for the generation of regular waves with $T_p = 1.67$s and $H_s = 0.05$m.

The wavemaker velocity $u(t)$ is then computed from the filtered motion $R(t)$ with a central-difference scheme, as

$$u(t) = \frac{R(t+\Delta t) - R(t-\Delta t)}{2\Delta t} + O(\Delta t^2), \tag{5.18}$$

with the measurement time step $\Delta t = 0.001$s. Figure 5.18 (middle row) shows the resulting estimated velocity. A verification is made against the measured wavemaker acceleration $a(t)$, by applying a central difference scheme to the estimated velocity. The time derivative of the estimated velocity and the measured acceleration are compared in Fig. 5.18 (bottom row), confirming that the velocity estimate is accurate. The wavemaker motion and velocity are sufficient to reproduce the measured wave dynamics in the numerical tank. The remainder of this chapter aims to validate the obtained numerical simulations on the aforementioned aspects .

5.6 Validation of the numerical wave tank

5.6.1 Capture and absorption of irregular waves

As explained in Chapter 4, a limit of existing absorbing-boundary methods such as those introduced by Peric and Maksoud [115, 116] is that they require an analytical solution for the waves, and may thus be used for the absorption of prescribed, regular waves only. The numerical tank and the coupling to the dry beach introduced in Chapter 4 respectively aim to simulate and absorb any type of waves, including irregular waves, without requiring the *a priori* estimation of the wave profiles. In this section, the numerical tank is validated against experimental measurements of irregular wave profiles.

Irregular waves with variable frequency, amplitude and steepness are considered. Their amplitude and frequency are computed randomly, with a peak period $T_p \approx 1.56$s and characteristic wave height $H_s = 0.1$m. A wave spectrum of 200s is generated from 100 equidistant frequency components between $\omega_1 = 3.5$rad/s and $\omega_2 = 4.5$rad/s, each with an amplitude $Z_a = H_s/2$, and a random phase $\epsilon_\omega = 2n\pi$ where n is a random number between 0 and 1:

$$\eta(t) = \sum_{\omega=3.5}^{4.5} Z_a \sin(\omega t + \epsilon_\omega). \tag{5.19}$$

To satisfy condition (5.11), phase angles of about nine waves are selected among the obtained spectrum. Three phase angles are considered hereafter: case 2.2.1, case 2.2.2 and case 2.2.3 (*cf.* Table 5.4). As $\omega \leq \omega_2 = 4.5\text{rad/s}$, the minimal wavelength is approximately $\lambda \approx 3.0\text{m}$. The deep-water spatial resolution is set to $\Delta x_{DW} = 0.05\text{m}$, so that the waves are discretised by at least 60 nodes. As a consequence, the shallow-water resolution is set to $\Delta x_{SW} = \Delta x_{DW}^2 = 0.0025\text{m}$. To satisfy the stability condition (3.71), the time step is set to $\Delta t = 0.001\text{s}$. Finally, the coupling point is set at $x_c = 28.24\text{m}$, which ensures sufficient water depth to guaranty stability of the potential-flow simulations.

First, we verify that the irregular waves are well captured by the numerical tank. To observe the wave-frequency spectra in each case, the numerical free-surface Fourier modes are compared to the experimental free-surface measurements in Fig 5.19 (case 2.2.1), D.7 (case 2.2.2, in Appendix D.1) and D.8 (case 2.2.3, in Appendix D.1). As expected, most of the frequency spectra are contained within $\omega_1 = 3.5\text{rad/s}$ and $\omega_2 = 4.5\text{rad/s}$. In the three cases, all the Fourier modes are captured by the numerical tank, thus confirming that irregular waves are accurately simulated in the numerical basin. The amplitude variation ($\leq 10\%$) is assumed to result from the aforementioned discrepancies between the experimental and numerical measurements (noise filtering, probe accuracy *etc.*).

To check whether the irregular waves are efficiently absorbed by the beach, the energy absorption is computed in each case, in Fig. 5.20. Figure 5.20 shows that above 98.9% of the energy is absorbed by the numerical beach. Therefore, the numerical tank is efficient in both the generation and absorption of irregular-wave profiles, despite the fixed-coupling location.

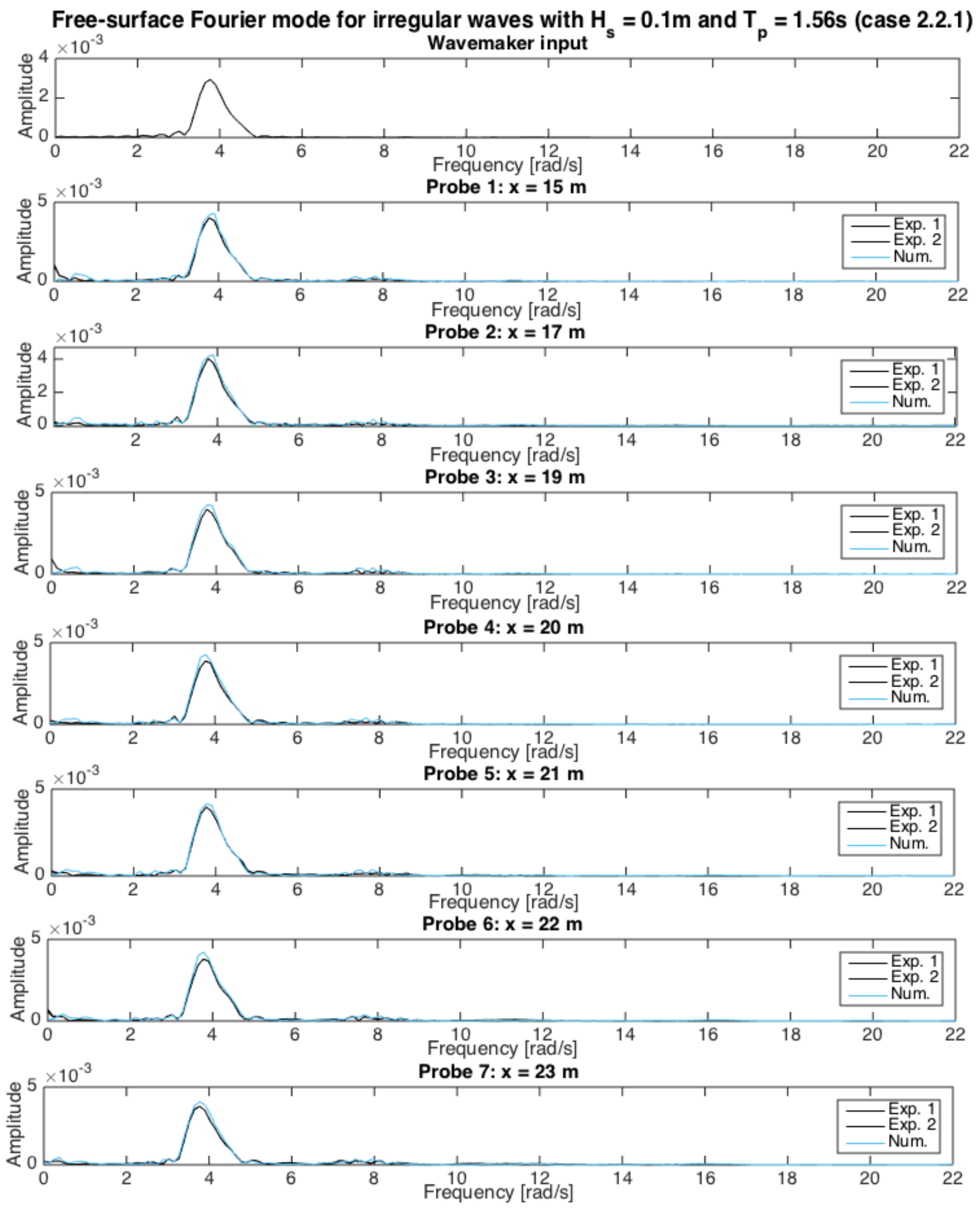


Figure 5.19: Comparison between numerical (blue) and experimental (black) Fourier modes of irregular waves measured at probes 1 to 7 with $H_s = 0.1$m and $T_p = 1.56$s (*cf.* wave profile in Table 5.4, case 2.2.1).

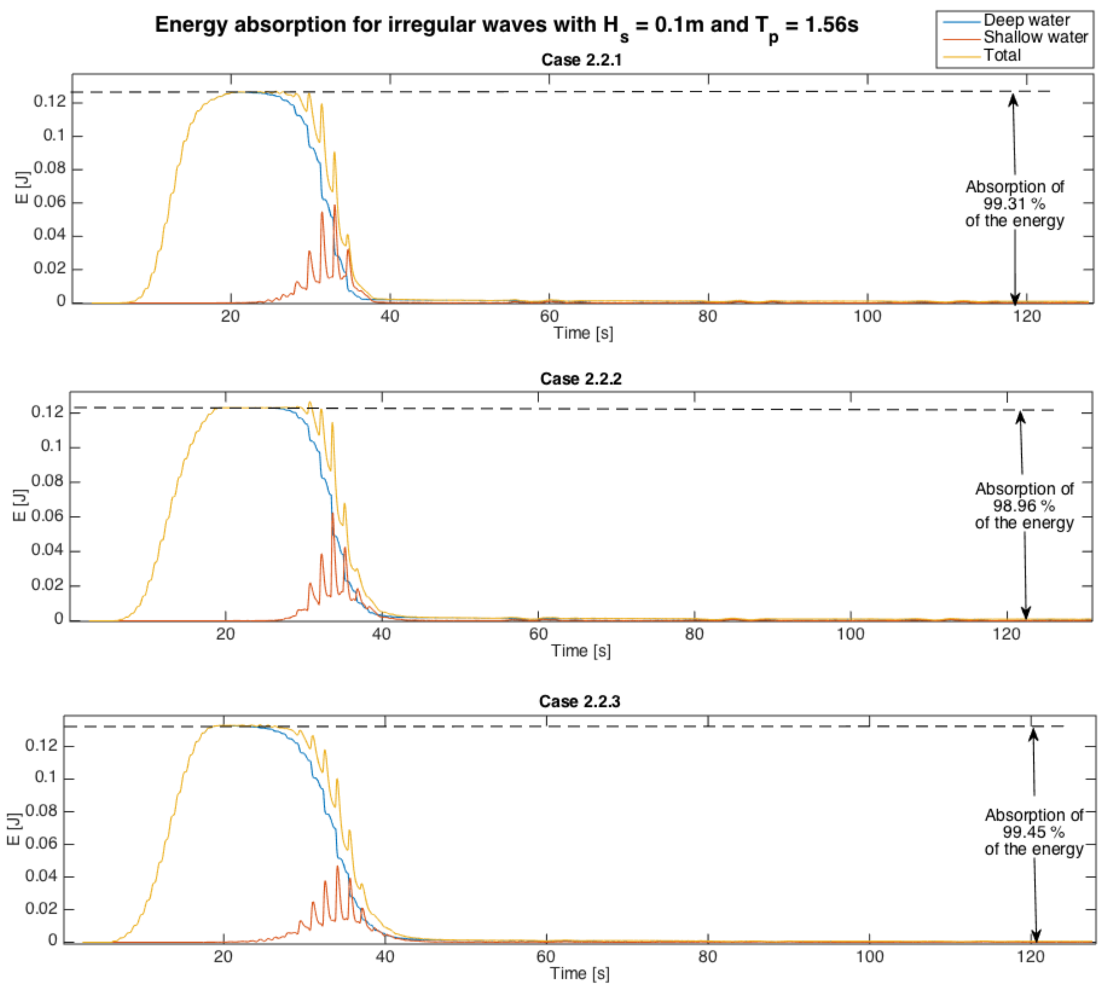


Figure 5.20: Energy absorption for irregular waves with $H_s = 0.1\text{m}$ and $T_p = 1.56\text{s}$. Three wave profiles are considered: case 2.2.1 (top row), case 2.2.2 (medium row) and case 2.2.3 (bottom row) (see Table 5.4).

5.6.2 Capture and absorption of long waves

In order to study the long-wave reflection of the 10%-slope beach, the long-wave case 1.1.1 (*cf.* Table 5.3) is repeated in the full-length tank, as shown in Fig. 5.21.

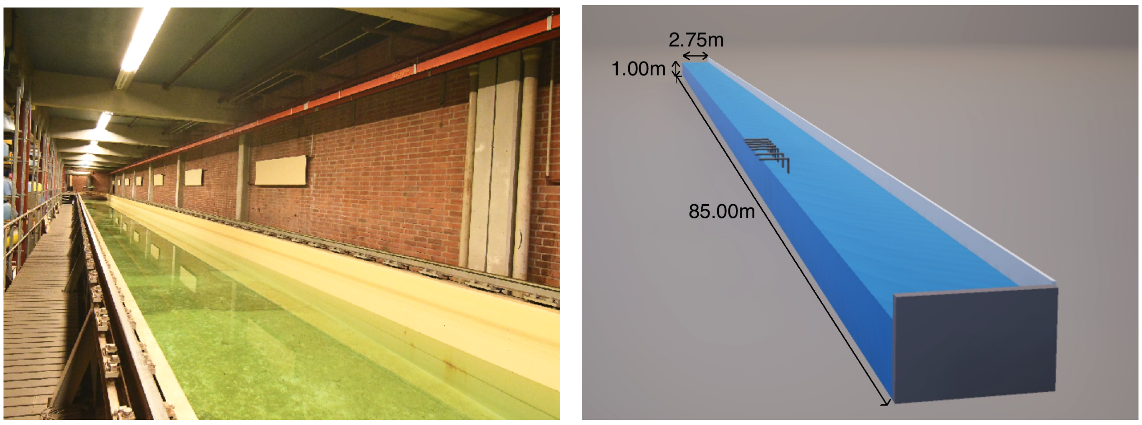


Figure 5.21: Photography (left) and schematic (right) of the experimental tank of TUD.

The objective is to compare the amplitude of the reflected waves in the tank-with-beach set-up and in the full-length tank, in order to evaluate the ability of the beach to both absorb and reflect long waves. In addition, simulations with $\Delta x_{DW} = 0.05$m, $\Delta x_{SW} = 0.0025$m and $x_c = 28.24$m are performed based on the measured wavemaker motion and velocity in the case with a beach.

Figure 5.22 compares the temporal evolution and the Fourier modes of the free-surface elevation in the case of experiments in the full-length tank (black), repeated experiments in the "beached" tank (purple) and simulations (blue). The top plot in Fig. 5.22 shows the temporal evolution of the free surface at probe 1 ($x_1 = 15m$) for the case of numerical data (blue), experimental data in the tank with a beach (purple) and experimental data in the full-length tank (black). Overall, the numerical waves agree with the "beach-tank" measurements, both in phase and amplitude. It is noteworthy that the comparison is better for the troughs of the waves than for the crests. As explained in section 5.3.3, this observation can result from a different accuracy along the probe depth, due to corrosion of the wire. The temporal evolution of the free surface is used to estimate the frequency components of the waves at various times.

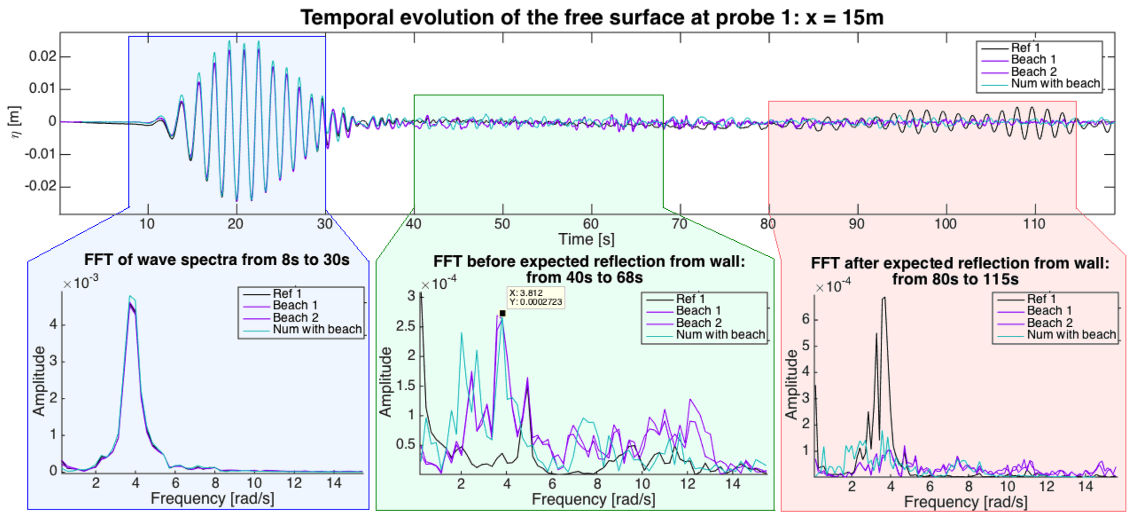


Figure 5.22: Wave profile at probe 1 ($x_1 = 15$m) are used to analyse the reflection of long waves at the beach and on a vertical wall. Top: temporal evolution of the free surface. Bottom left: Fourier modes of the wave spectrum from 8s to 30s revealing the main-frequency components of the spectrum. Bottom middle: Fourier modes of the wave profile from 40s to 68s to highlight reflection of long waves on the 10%-slope beach. Bottom right: Fourier modes of the wave profile from 80s to 115s to obtain the reflection of long waves on a vertical wall.

First, the main wave-frequency components are evaluated by computing the Fourier modes of the measured and numerical wave profiles between $t = 8$s and $t = 30$s (blue area). In Fig. 5.22, in which the amplitude of each frequency component is given in the bottom-left subfigure, a regular wave spectrum with main frequency components contained between $\omega = 3.70\text{rad/s}$ and $\omega = 3.99\text{rad/s}$ can be observed, which agrees with the intended wave frequency $\omega = 3.76\text{rad/s}$ from Table 5.3. The time needed by the waves to travel to the end of the full-length tank (that is, along a distance of $L = 85.0$m) and to be reflected back to probe 1 at $x_1 = 15.0$m can be calculated from the main wave frequency; waves of frequency $\omega = 3.70\text{rad/s}$ travel at speed $c = 2.42\text{m/s}$, thus taking

$$t = (2L - x_1)/c \approx 64 \text{ s} \tag{5.20}$$

to travel back to probe 1. As there is a delay of approximately 5s recorded before starting the wavemaker, only waves reflected from the beach should be measured in the time interval $[40\text{s}, 68\text{s}]$.

The Fourier modes of the wave signal between $t = 40$s and $t = 68$s can be observed in the middle of the bottom subfigure of Fig. 5.22. As expected, the main frequency component of both the numerical and experimental measurements in the case with a beach is $\omega = 3.81\text{rad/s}$, thus corresponding to the waves reflected on the 10%-slope beach. This observation confirms that the beach partially reflects the long waves. By comparison with the bottom-left subfigure, for which the length of the time interval was similar, the portion of reflected waves is less than one tenth of the initial wave spectrum (amplitude of about 2.7×10^{-4} against 4.8×10^{-3} on the bottom-left plot is Fig. 5.22). However, the reflection factor must be evaluated with care since the amplitudes of the frequency components depend highly on the time interval considered. Another major observation is that, contrary to the case with beach or to the numerical data, the amplitude of the long-wave frequency component is negligible in the case of the full-length tank. As expected, waves with frequency $\omega \approx 3.8\text{rad/s}$ have not yet travelled back to probe 1, meaning that the full-length tank still behaves as a transparent boundary during the time interval [40s,68s]. Frequencies larger than 4.7rad/s are assumed to result from noise, as they are measured in both the full-length and "beached" tanks. Finally, a major frequency component between 0.1 and 0.5rad/s can be observed in the case of the full-length tank. These frequencies correspond to extremely-long waves ($\lambda > 40$m) that require hours to be dampened. These measurements are thus assumed to result from remaining long waves in the basin before the start of the measurements. The behaviour of the black curve in the top subfigure of Fig. 5.22 tends to confirm this assumption: the free-surface level decreases before any wave has reached probe 1 (for $t < 10$s), and the long-wave noise may be observed all along the evolving profile.

Finally, the bottom-right subfigure of Fig. 5.22 confirms that the reflection from the beach is negligible compared to the reflection at the end of the full-length basin. Most of the frequency components measured in the time interval [80s,115s] in the full-length tank correspond to waves generated with frequency $\omega \approx 3.8\text{rad/s}$. The deep-water part of the basin is therefore much more disturbed by reflected waves than the tank with beach. The red area in the top figure confirms that the reflected-wave amplitudes in the full-length tank are non-negligible compared to the initial wave profiles, while Fig. 5.23 shows that reflected waves in the "beached" tank correspond to less than 2% of the initial energy.

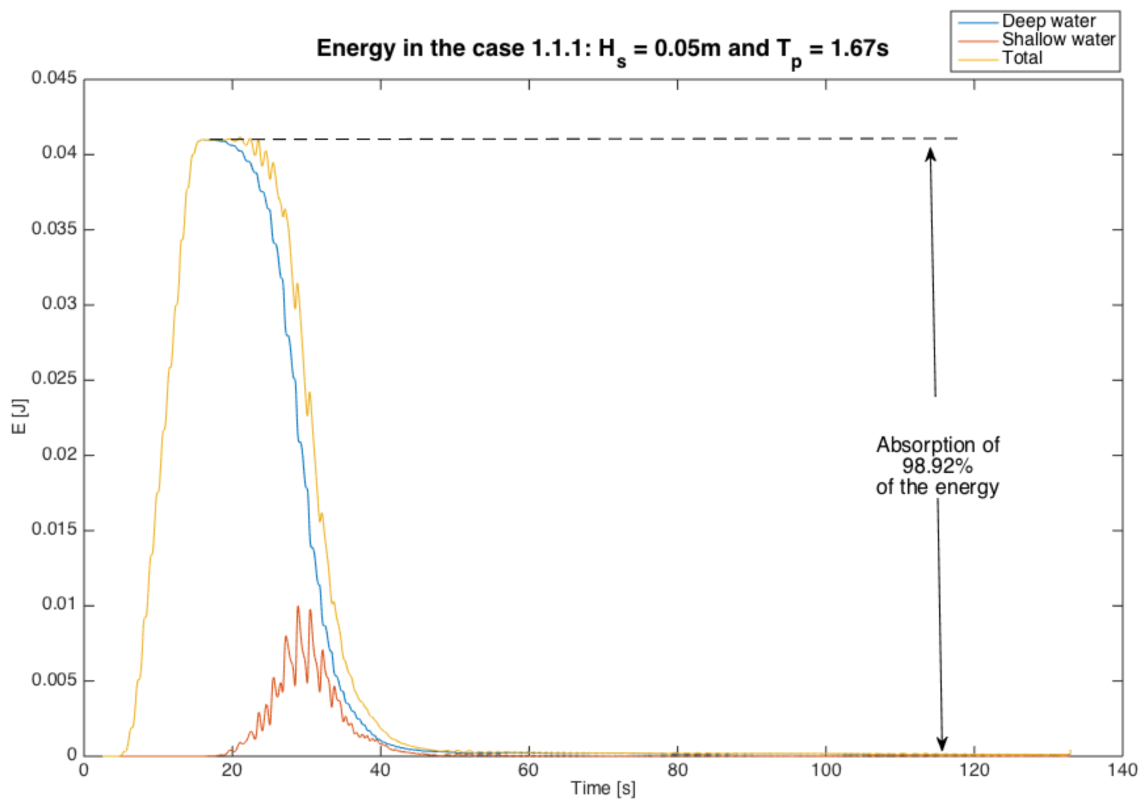


Figure 5.23: Numerical energy in the "beached" tank for case 1.1.1 (see Table 5.3) with $H_s = 0.05m$, $T_p = 1.67$s and $x_c = 28.24$m.

These results confirm that the addition of a beach in the numerical domain considerably reduces the computational time, since even in a domain of almost three times the length of the "beached" tank (85m as opposed to 30m), reflected waves rapidly disturb the deep-water area. Figures 5.22 and 5.23 also indicate that a beach with slope 10% is particularly efficient to absorb the waves, including long waves that are expected to partially reflect. Other, steeper, slopes could be considered to measure the reflection of long waves.

5.6.3 Capture and absorption of short waves

In the numerical tank derived in this thesis, waves are generated from a piston wavemaker. As explained in the conclusions of Chapter 3, the circular particle motion of deep-water waves may not be accurately generated from a uniform height-independent horizontally-translating wavemaker motion. The generation of short waves with a piston wavemaker results in a modulation of the wave amplitude, thus disturbing the expected free-surface deviation. However, to test wave-structure interactions in experimental wave tanks, it is important for engineers and designers to predict as accurately as possible the profile of the waves that will impact the structures in order to avoid over- or under-estimation of the wave forces. Similarly, when studying the factors resulting in a freak wave in a target area, one must ensure that no adverse effect will disturb the extreme-wave generation. The incapacity of the piston wavemaker to simulate realistic short waves thus limits the use of piston wavemakers installed in experimental tanks, which are nonetheless expensive and sometimes difficult to replace by more accurate flap-type wavemakers. A model that can simulate the amplitude modulation resulting from the piston motion would therefore be of great interest to the maritime industry, and in particular to MARIN, to predict and accommodate the disturbance caused to the free-surface deviation. The aim of this section is therefore to validate the ability of the numerical tank to capture the dynamics of short, deep-water waves generated by a piston wavemaker.

Waves are considered as short waves, or deep-water waves, when their wavelength is less than or equal to twice the water depth, that is, when

$$\lambda \leq 2H_0. \tag{5.21}$$

Since the water depth at rest in the flat experimental tank is $H_0 = 1.0$m, regular waves with $H_s = 0.03$m and wavelength $\lambda \approx 1.0$m are generated (case 1.3.1 in Table 5.3), thus satisfying the deep-water criterion (5.21). The coupling interface is set at $x = 28.24$m, with deep-water resolution $\Delta x_{DW} = 0.05m$ and $\Delta x_{SW} = \Delta x_{DW}^2 = 0.0025$m.

To observe the amplitude modulation resulting from the piston motion, the evolving profiles of the experimental and numerical free-surface elevations at the probes WHM1 to WHM7 are compared in Fig. 5.24 . To reduce the measurement errors, the experimental measurements are repeated and denoted by *Exp. 1* and *Exp. 2*. Figure 5.24 shows that the short-wave profiles are well captured by the deep-water model. As intended (see Table 5.3), both the experimental and numerical wave heights are about $H_s = 0.03$m, thus confirming the accuracy of the transfer function computed in section 5.5.1.

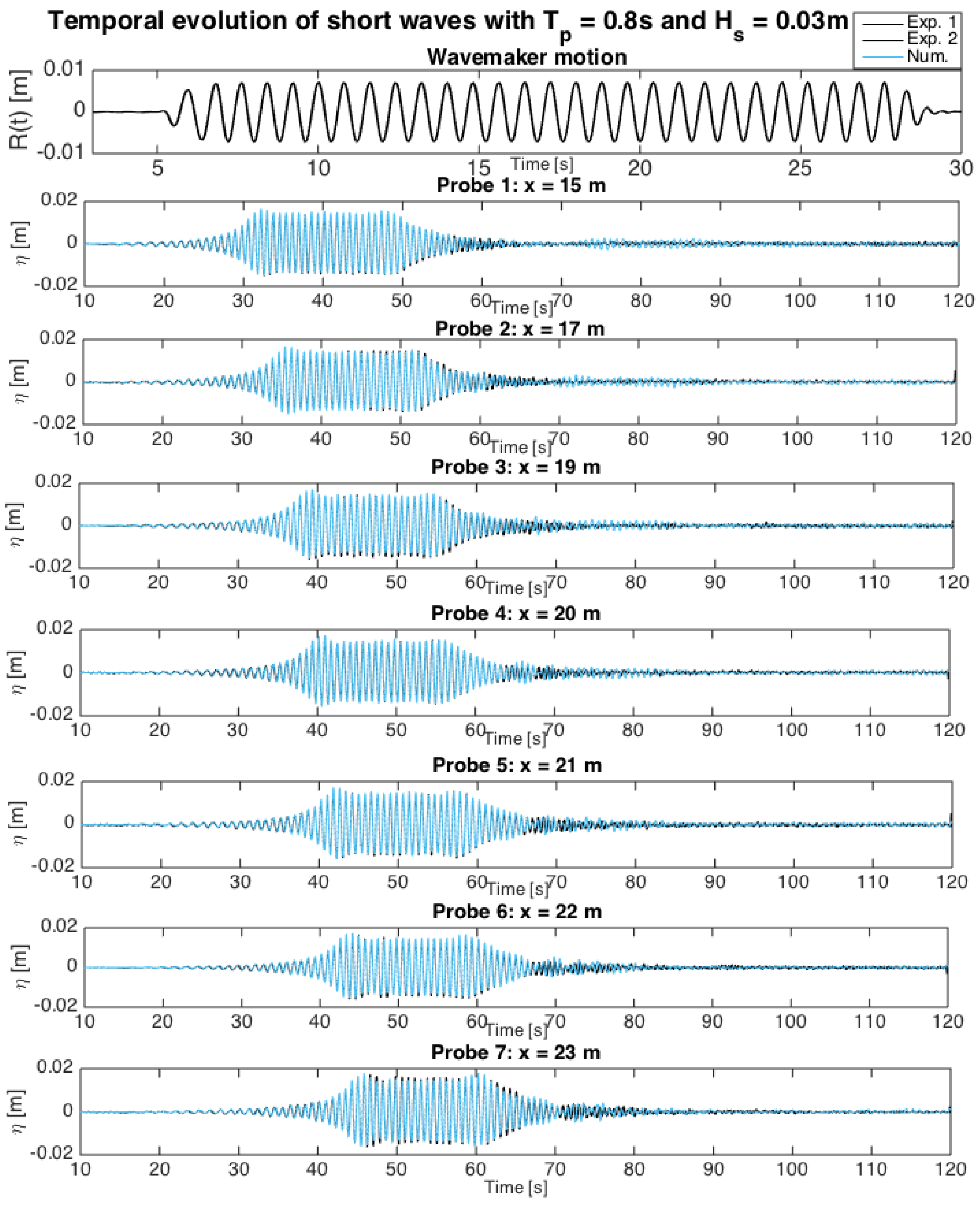


Figure 5.24: Temporal evolution of the free surface at probes WHM1 to WHM7 in the case 1.3.1, that is, for $T_p = 0.80$s and $H_s = 0.03$m (see Table 5.3). The coupling point is set at $x_c = 28.24$m, yielding a coupling-depth at rest of $H_c = 0.2$m.

While the wavemaker amplitude is regular, the numerical and experimental wave profiles have peaks of amplitude, thus confirming that the numerical tank may be used to predict the experimental short-wave profiles and to avoid unexpected amplitude modulation. However, some disturbances of the numerical signal can be observed after the main wave profile; while the experimental measurements mainly consist in noise measurements, the numerical signal contains some low-amplitude waves that can result from either an instability at the coupling point or a partial reflection at the beach.

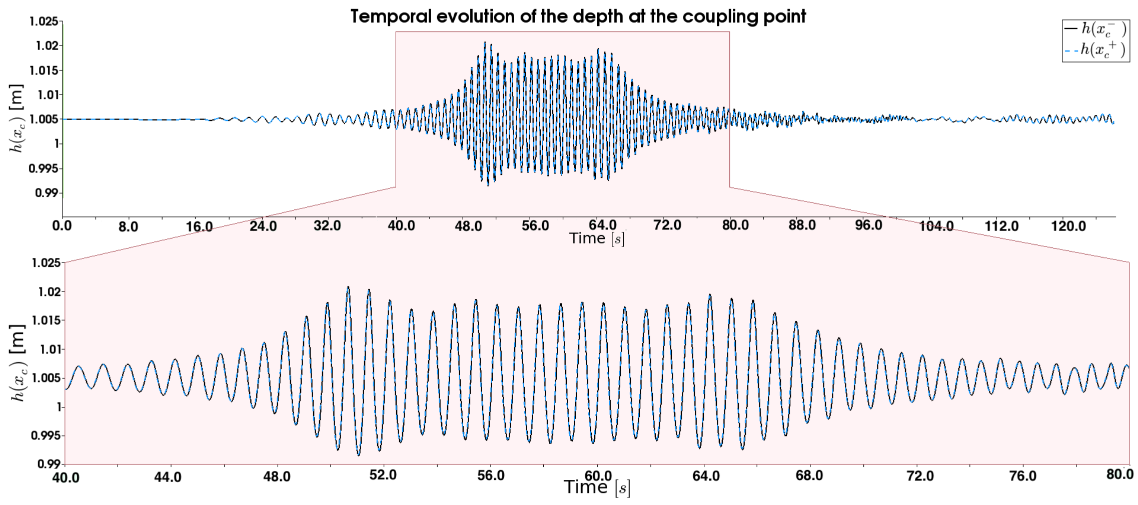


Figure 5.25: Temporal evolution of the deep- (black) and shallow-water (dashed blue) depths at the coupling point $x_c = 28.24$m for the case 1.3.1 (see Table 5.3).

To eliminate the first option (*i.e.*, discontinuity at the coupling point resulting in some instability) the temporal evolutions of the deep-water and shallow-water depths at $x = x_c$ are compared in Fig. 5.25. The top subfigure shows their evolution over the whole time interval $[0\text{s}, 120\text{s}]$, while the second subfigure focuses on the main wave profile in which waves are higher and steeper. The deep- and shallow-water depths at the coupling point agree during the whole time interval, including when the short waves cross the coupling interface, thus confirming that the coupling is continuous and stably captures the short-wave transfer from deep to shallow water. Therefore, the reflected waves noticeable in Fig. 5.24 probably result from reflection of the main wave signal on the beach.

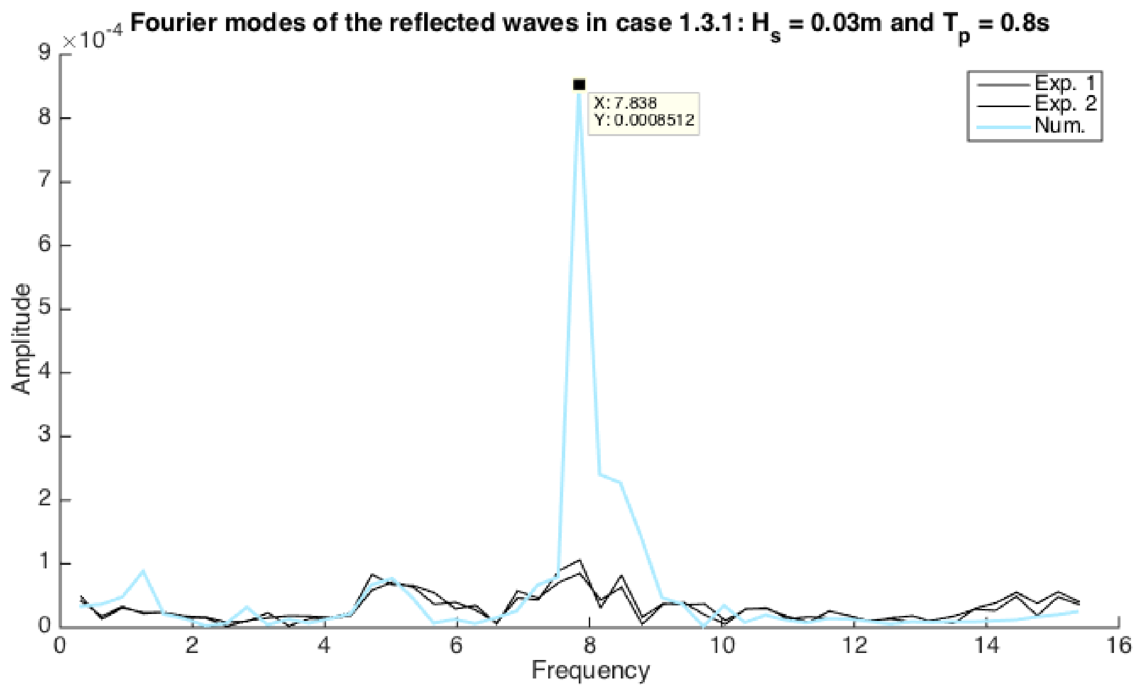


Figure 5.26: Fourier modes of the numerical (blue) and experimental (black) reflected waves measured at probe 1 of case 1.3.1 ($H_s = 0.03$m and $T_p = 0.8$s, see Table 5.3) during the time interval $[70\text{s}, 90\text{s}]$. The coupling is set at $x_c = 28.24$m.

To confirm this assumption, the Fourier modes of the numerical free-surface elevation at probe 1 are computed for the time interval during which reflected waves are observed, that is, for $t \in [70\text{s}, 90\text{s}]$ (*cf.* top-right subfigure of Fig. 5.24). The results are displayed in Fig. 5.26, together with the Fourier modes of the experimental free-surface elevation during the same time interval. As expected, the waves measured in the time interval $[70\text{s}, 90\text{s}]$ have a frequency of $\omega = 7.84\text{rad/s}$, which is the frequency of the waves initially generated by the wavemaker (*cf.* Table 5.3). Therefore, these waves do not result from any instability in the tank but from partial reflection at the beach. This observation indicates that the coupling is able to continuously and stably transfer short waves from deep to shallow water (and vice-versa) but that improvements can be made to the beach model so that it absorbs these short waves more accurately. The shallow-water model accuracy is higher for shallow-water waves, that is, waves satisfying $\lambda \geq 20H$. In the case presented in this section, the coupling was set to $x_c = 28.24$m, with rest depth

$H(x_c) = 0.2$m$= \lambda/5$, so the shallow-water limit was not satisfied. One option to increase the efficiency of the shallow-water beach model is to shift the coupling point to shallower water. However, one must ensure that the depth is deep enough to avoid wave breaking. Since the breaking-wave limit for waves with $\lambda = 1.0$m is at $h \leq \lambda/20 = 0.05$m, the depth at the coupling point is set to $H(x_c) = 0.08$m to ensure that $h - H_s > \lambda/20$ and hence that wave breaking in the deep-water domain is avoided.

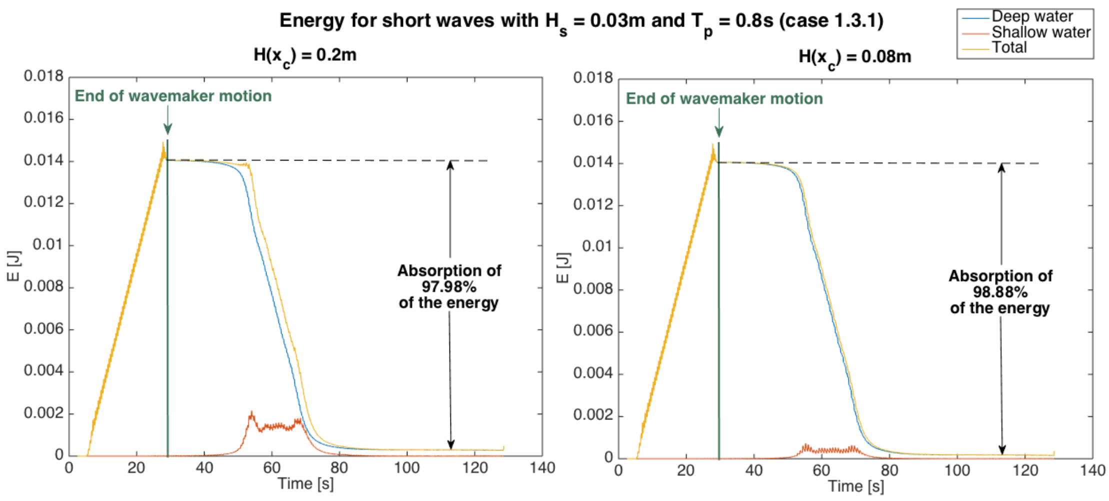


Figure 5.27: Numerical energy in the case 1.3.1: short waves with $H_s = 0.03m$ and $T_p = 0.8$s. Left: coupling at $x_c = 28.24$, with $H(x_c) = 0.2$m$= \lambda/5$. Right: coupling at $x_c = 29.44$m, with $H(x_c) = 0.08$m$= \lambda/12.5$.

Figure 5.27 compares the energy of case 1.3.1 (see Table 5.3) when the coupling is at $x_c = 28.24$m (left) and $x_c = 29.44$m (right). In the first simulation, with $h(x_c) \approx 0.2$ m $= \lambda/5$, the beach absorbs 97.98% of the wave energy and is thus efficient despite the partial free-surface disturbance observed in Fig. 5.24. However, as expected, the energy absorption is increased by shifting the coupling interface in shallower water, at $x_c = 29.44$m where $h(x_c) \approx 0.08$m $< \lambda/12$. In that case, the beach absorbs 98.88% of the energy. Figure. 5.28 compares the amplitude of the frequency modes of the two simulations with the experimental measurements for $t \in [70\text{s}, 90\text{s}]$, that is, when only reflected waves are measured. The amplitude of the reflected waves has dropped from 8.512×10^{-7} for the deepest coupling where $h(x_c) = 0.2$m (red) to 5.050×10^{-7} for shallower coupling where $h(x_c) = 0.08$m (green);

that is, a decrease of 40%. This solution can thus be used to reduce short-wave reflection by the numerical beach, but is limited by the breaking-wave limit, which may quickly be reached for high-amplitude waves. In section 5.6.4, we show that the beach is indeed not optimal to absorb steep, short waves.

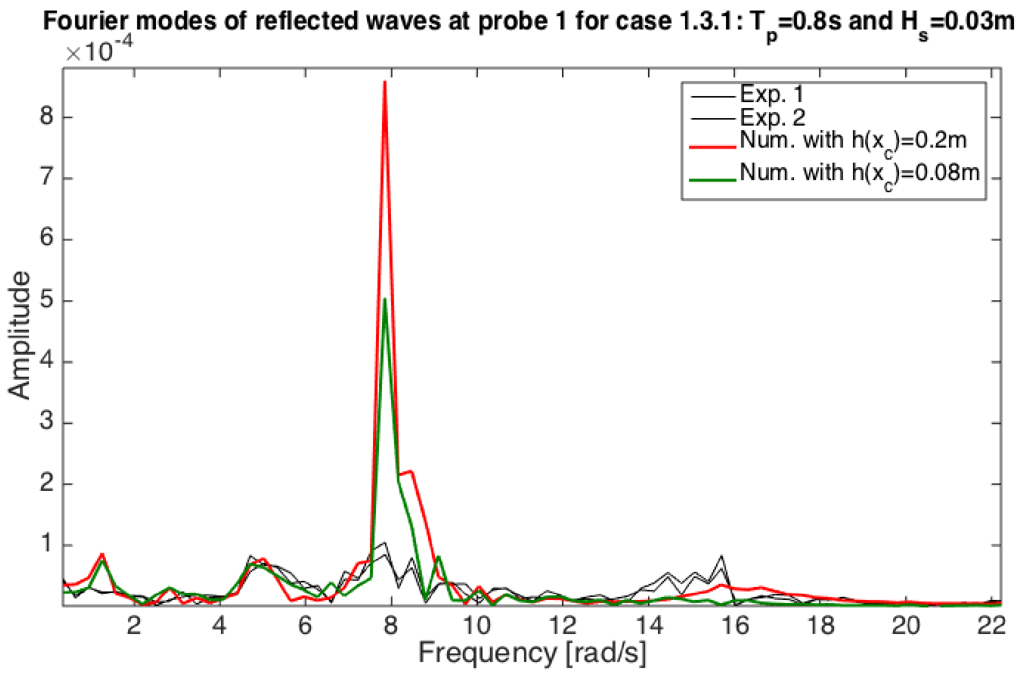


Figure 5.28: Experimental (black) and numerical (red and green) frequency spectra of the free surface at probe 1 ($x_1 = 15$m) for the case 1.3.1 (see Table 5.3) to highlight the effect of the coupling location on the wave reflection. The case where $h(x_c) = 0.08$m (green) reflects the waves 40% less than the case where $h(x_c) = 0.2$m (red).

5.6.4 Capture and absorption of steep waves

A main objective of this thesis is to enable the generation of rogue-type waves in a target area of the experimental or numerical wave tank. Due to their great steepness, these extreme waves can be captured only by a nonlinear model, hence the necessity to implement a nonlinear absorbing boundary through the shallow-water beach. The numerical tank presented in Chapter 4 is the first fully nonlinear coupled model, and should therefore be able to absorb steep, nonlinear waves. The purpose of this section is to evaluate the ability of the numerical model to absorb steep waves approaching the breaking-wave limit.

The steepness of a wave is hereafter defined as the ratio between its wave height H_s and wavelength λ, that is,

$$s_w = \frac{H_s}{\lambda}. \tag{5.22}$$

The blue curve in Fig. 5.29 shows the maximum wave height that the wavemaker is able to generate in the frequency range $\omega \in [2.53, 7.91]$ and may be understood as the breaking-wave limit, for which $s_w = 0.05$. To analyse the ability of the numerical tank to simulate and absorb steep waves, waves at the breaking-wave limit are generated in the case of three different frequencies, denoted by the red squares in Fig. 5.29 and whose parameters are specified in Table 5.8. By comparing these three cases of equal steepness but differing wavelength and wave amplitude, we aim to validate the accuracy of the numerical wave tank for the simulation, the coupling and the absorption of steep waves in various configurations.

Case number	ω [rad/s]	H_s [m]
1.1.3	3.76	0.2
1.2.3	5.56	0.1
1.3.2	7.85	0.05

Table 5.8: Frequencies and wave heights of the steep-wave profiles of cases listed in Table 5.3.

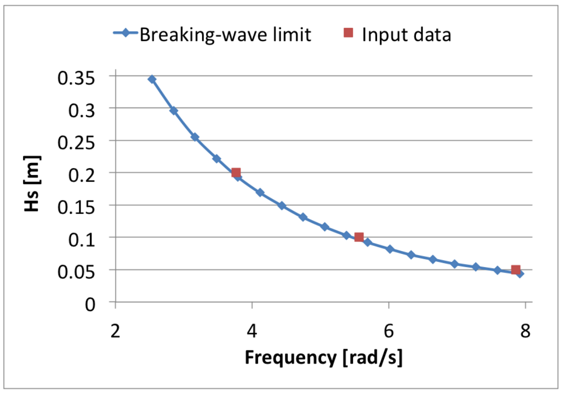


Figure 5.29: Specification of steep waves from the breaking-wave limit.

In each case, the coupling location is defined from the wavelength and amplitude. For case 1.1.3, with wavelength $\lambda \approx 4.0$m, the shallow-water limit, which leads to wave breaking, is reached when $h(x) = 0.2$m. The coupling point is thus set to satisfy $H(x_c) = 0.4$m, so that the minimal depth in the deep-water domain is $h(x_c) \approx H(x_c) - H_s/2 \approx 0.3\text{m} > \lambda/20$. By the same rationale, the coupling points in cases 1.2.3 and 1.3.2 are both set to $x_c = 27.24$m so that $H(x_c) = 0.3$m. In the three cases, the deep-water resolution is set to $\Delta x_{DW} = 0.05$m, while the shallow-water resolution is set to $\Delta x_{SW} = 0.0025$m for cases 1.2.3 and 1.3.2, and to $\Delta x_{SW} = 0.004$m for case 1.1.3.

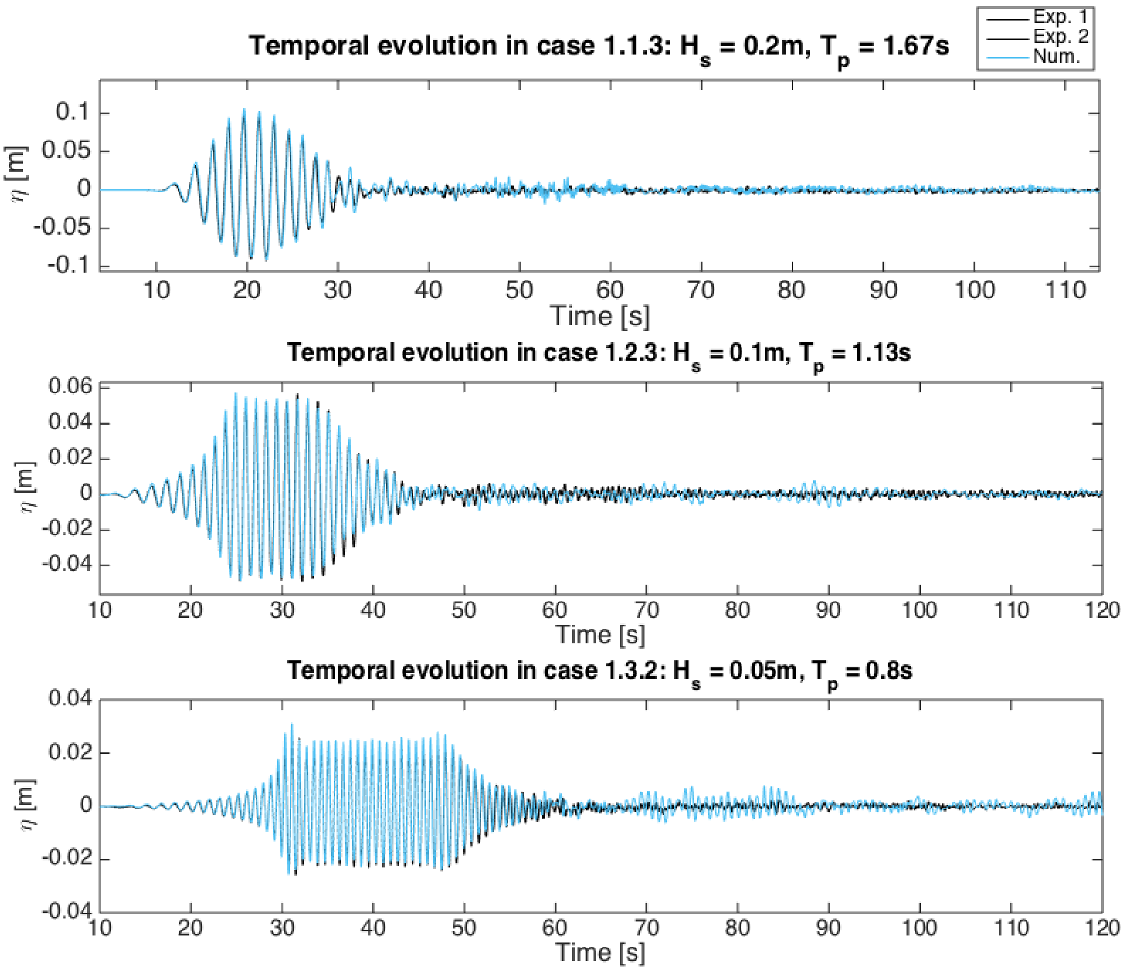


Figure 5.30: Numerical (blue) and experimental (black) evolution of the free-surface elevation at probe 1 when generating steep waves approaching the breaking-wave limit (case 1.1.3 (top), 1.2.3 (middle) and 1.3.2 (bottom), see Table 5.3).

To validate the simulation of steep waves by the deep-water model, the temporal evolution of the numerical and experimental free-surface elevation at probe 1 are compared in Fig. 5.30 for each case 1.1.3 (top), 1.2.3 (middle) and 1.3.2 (bottom). In the three cases, the steep waves are well captured by the potential-flow model, which indicates that the coupling to shallow water enables wave-breaking to be accommodated by the shallow-water model, thus ensuring stability of the simulations despite waves approaching the breaking-wave limit.

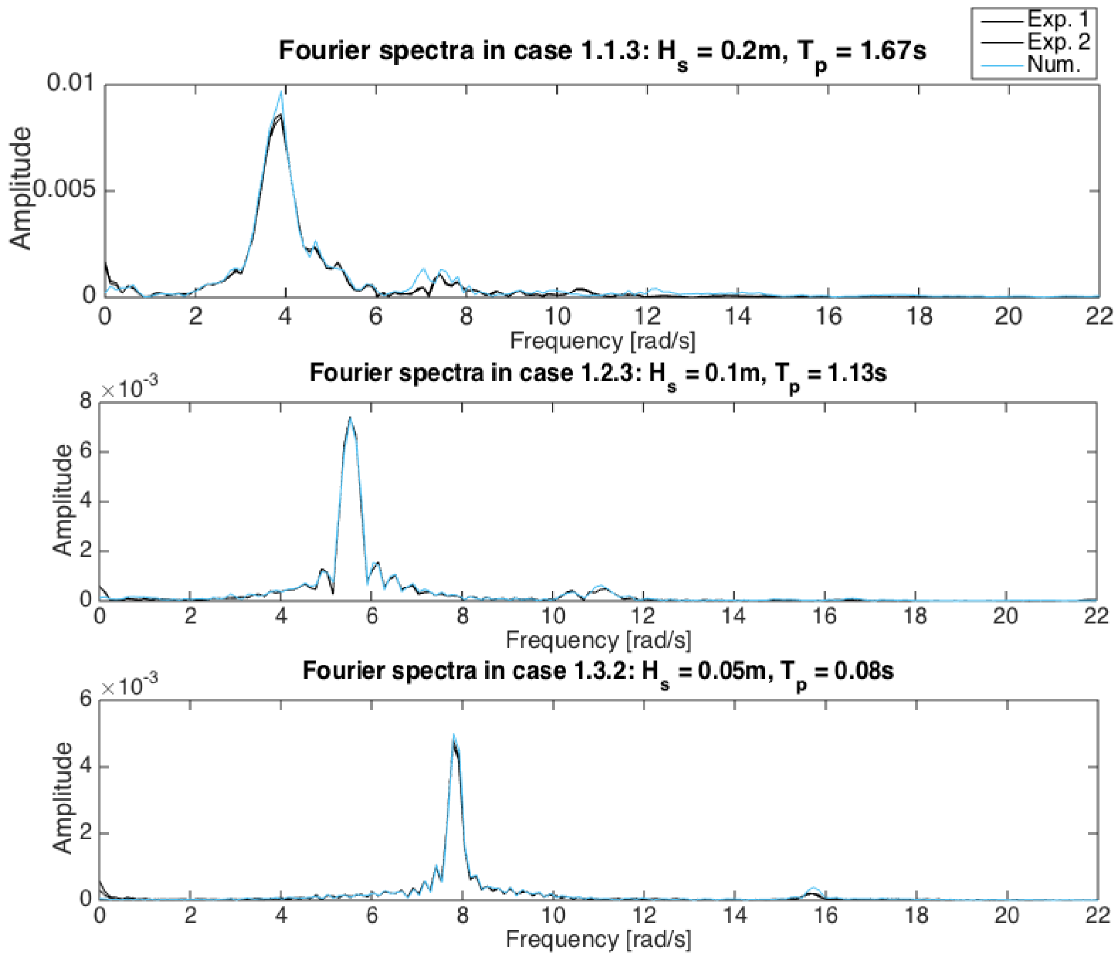


Figure 5.31: Numerical (blue) and experimental (black) Fourier spectra of the free-surface elevation at probe 1 when generating steep waves approaching the breaking-wave limit (case 1.1.3 (top), 1.2.3 (middle) and 1.3.2 (bottom), see Table 5.3).

Figure 5.31 compares the experimental (black) and numerical (blue) Fourier wave spectra in all three cases. The agreement between the numerical and experimental frequency modes in all three cases indicates that no extra frequency is detected in the numerical model, thus confirming that the simulations are stable. This observation confirms the efficacy of the numerical tank in dealing with steep waves, which was not the case for the 3D potential-flow model of Chapter 3.

In particular, the coupling process is continuous despite the steepness of the waves, as shown in Fig. 5.32. In all three cases, the deep- and shallow-water depths are equal at the coupling point, and, most importantly, their evolution does not show any unstable behaviour, thus confirming the nonlinearity of the coupling. The coupled model can therefore be used to simulate steep waves and test rogue-wave impact on maritime structures. However, its potential for utilisation may transpire to be limited because of the fixed coupling point, which must be set with precaution to ensure both stability of the potential-flow model and validity of the shallow-water assumption. As shown in Fig. 5.30, despite being well captured by the deep-water model, part of the steep waves is reflected by the shallow-water beach for all three cases.

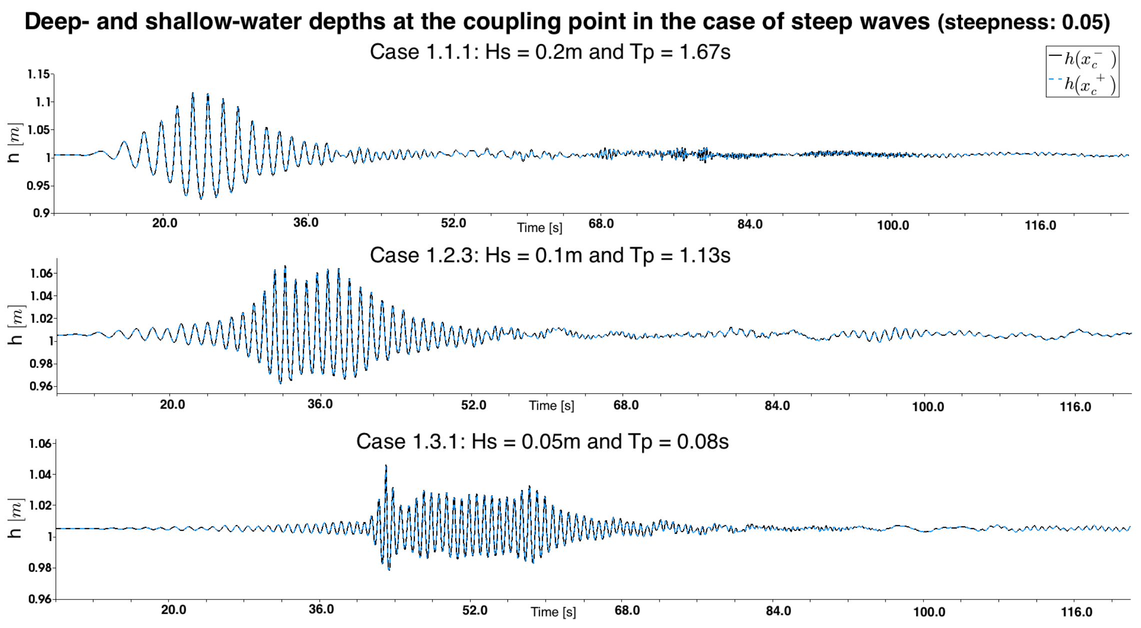


Figure 5.32: Deep- and shallow-water depth at the coupling point for case 1.1.3, 1.2.3 and 1.3.2 in which steep waves approaching the breaking-wave limit are simulated (see Table 5.3).

Figure 5.33 shows that this reflection is still much lower than reflection from a rigid wall. The energy of the three cases confirms that, for a given steepness, the simulation of long waves is more accurate than in the case of short waves. This observation is consistent since short waves require coupling at shallow-water depth (*cf.* results of section 5.6.3) while steep waves require coupling at deep-water depth so that waves do not approach the breaking-wave limit. Dealing with these two constraints requires finding a compromise between capture (deep coupling) and absorption (shallow coupling) of the waves.

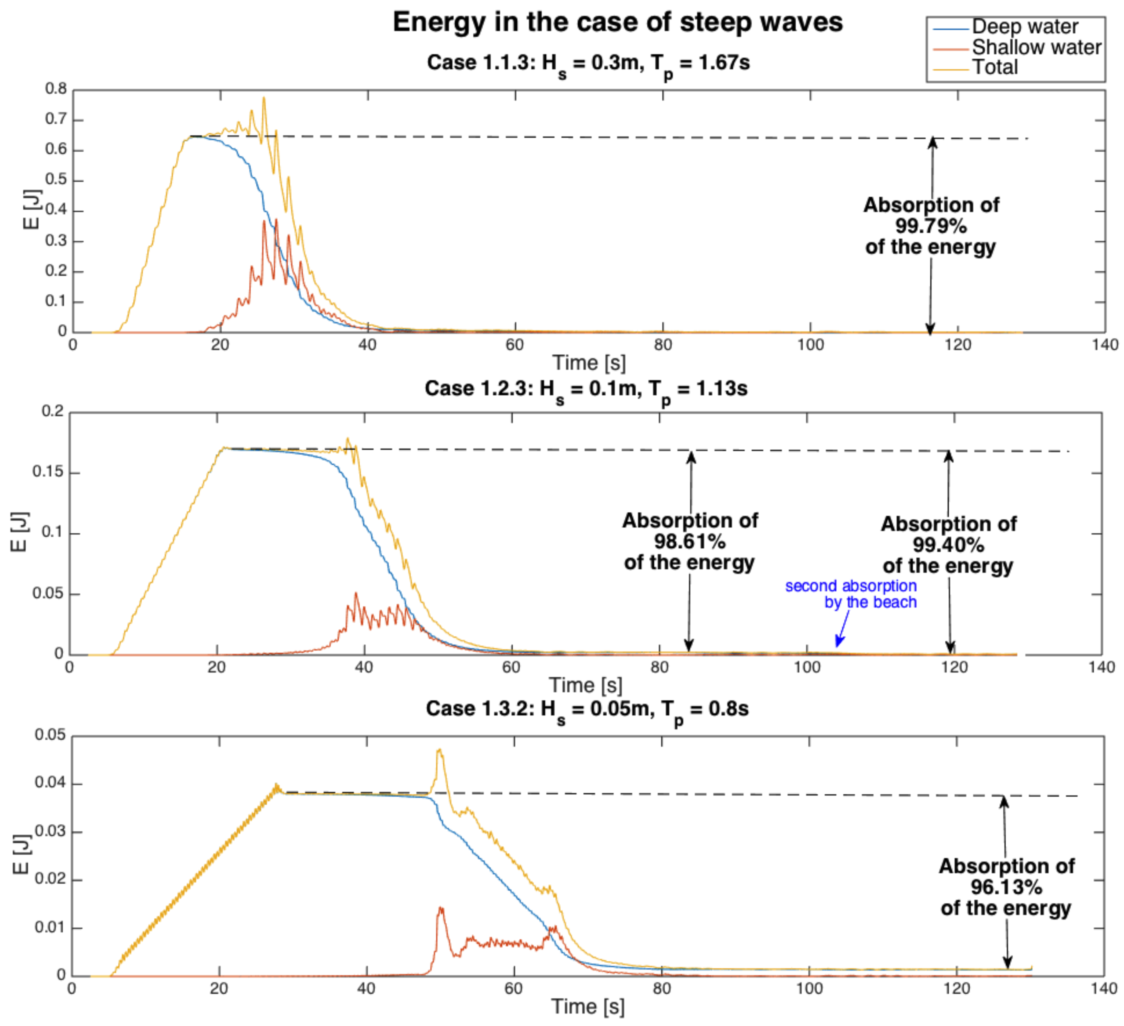


Figure 5.33: Numerical deep- and shallow-water energy for case 1.1.3, 1.2.3 and 1.3.2 in which steep waves approaching the breaking-wave limit are simulated (see Table 5.3).

5.7 Conclusion

The experiments conducted at TUD have enabled the validation of the accuracy of the numerical tank for the simulation, the coupling and the absorption of various types of waves.

First, the deep-water part of the numerical tank shows good accuracy for the simulation of irregular, regular, long, short and steep waves, both in terms of amplitude and frequency. In particular, it can be used by the maritime industry to predict amplitude modulation resulting from the translation of the piston wavemaker during the generation of short waves in the experimental basins.

Second, the coupling process developed in Chapter 4 is continuous and stable for all types of waves, including steep and irregular waves. The first nonlinear coupling process developed in Chapter 4 can therefore be used to stably couple other systems of nonlinear equations with the variational approach. In addition, by adjusting the position of the coupling interface depending on the wavelength and amplitude, good agreement between our numerical simulations and the experimental measurements was obtained for wave steepness up to the breaking-wave limit. Therefore, the coupling between the finite-element method and the finite-volume method enables the capture of waves from their generation at the wavemaker to their absorption at the beach, and, in particular, wave propagation in deep water as well as wave breaking at a shallow-water beach. The coupling of the two models and numerical schemes is therefore an efficient method to model water-wave dynamics. However, an intrinsic choice of the coupling location, as in [33, 55], would optimise the efficiency of the coupling, in particular for short waves for which the balance between stability of the deep-water model and accuracy of the shallow-water model is very sensitive to the wave amplitudes. The coupling location must indeed be set depending on the waves' profiles in order to satisfy both the deep-water assumption (that is, waves' amplitudes are sufficiently small relative to the water depth) and the shallow-water limit (that is, the velocity profile is depth-independent).

Finally, the 10%-slope beach shows efficient absorption of the waves, both experimentally and numerically. However, the experimental reflection factor of long waves with the 10%-slope beach could not be measured because in this case the reflected waves were negligible. Another slope should be tested, both experimentally and numerically, to develop further conclusions on the reflection of long waves. The numerical-beach-model absorption improves considerably the computational performance of the deep-water model of Chapter 3. Future work includes the extention to more accurate wave-breaking and beach models, so that the numerical tank can be used to test breaking-wave impact on structures. Measurements of the waterline and breaking location conducted at TUD are available for validation, and potential optimisation, of future extensions of beach models.

In addition to the experimental data recorded for the validation presented in this chapter, detailed tutorials of the implementation of the models are explained in Chapter 6, thus allowing not only the use but also further extension of the present numerical tank.

Chapter 6

Code tutorials

6.1 Introduction

The models derived in this thesis were built to address industrial and academical challenges. To facilitate the use and the extension of the models, detailed tutorials on how to both use and extend the codes are presented in this Chapter. A folder containing all the codes is made available together with this thesis. Figure 6.1 shows the code source of the main folder, named *Variational water-wave models*. This folder contains a *Benney-Luke* folder, in which the Benney-Luke model for oblique intersection of solitary waves presented in Chapter 2 is implemented. Details on how to use and to implement this code are given in section 6.2. It also contains a folder *Numerical tank* in which one can choose between solving the potential-flow equations in the deep-water tank, as presented in Chapter 3, or the coupled equations presented in Chapter 4, in the coupled tank. The description of the implementation strategies used in both cases is given in sections 6.3 and 6.4 respectively. Finally, in section 6.5, a tutorial on how to import the data collected at Delft University of Technology (DUT) is given in order to allow validation of future extensions of the coupled model, as suggested in Chapter 5. Each section of this chapter contains a presentation of the model, a description of the code source, a list of the solutions saved during the simulations, a tutorial to run the code, and details of the code implementation. The finite-element implementation of the models is eased by the use of Firedrake [120, 9, 7, 71, 96], that can be installed following the instructions given in the link: `https://www.firedrakeproject.org/download.html`. The mesh generator Gmsh [56] is also required to build the mesh used in section 6.2 for

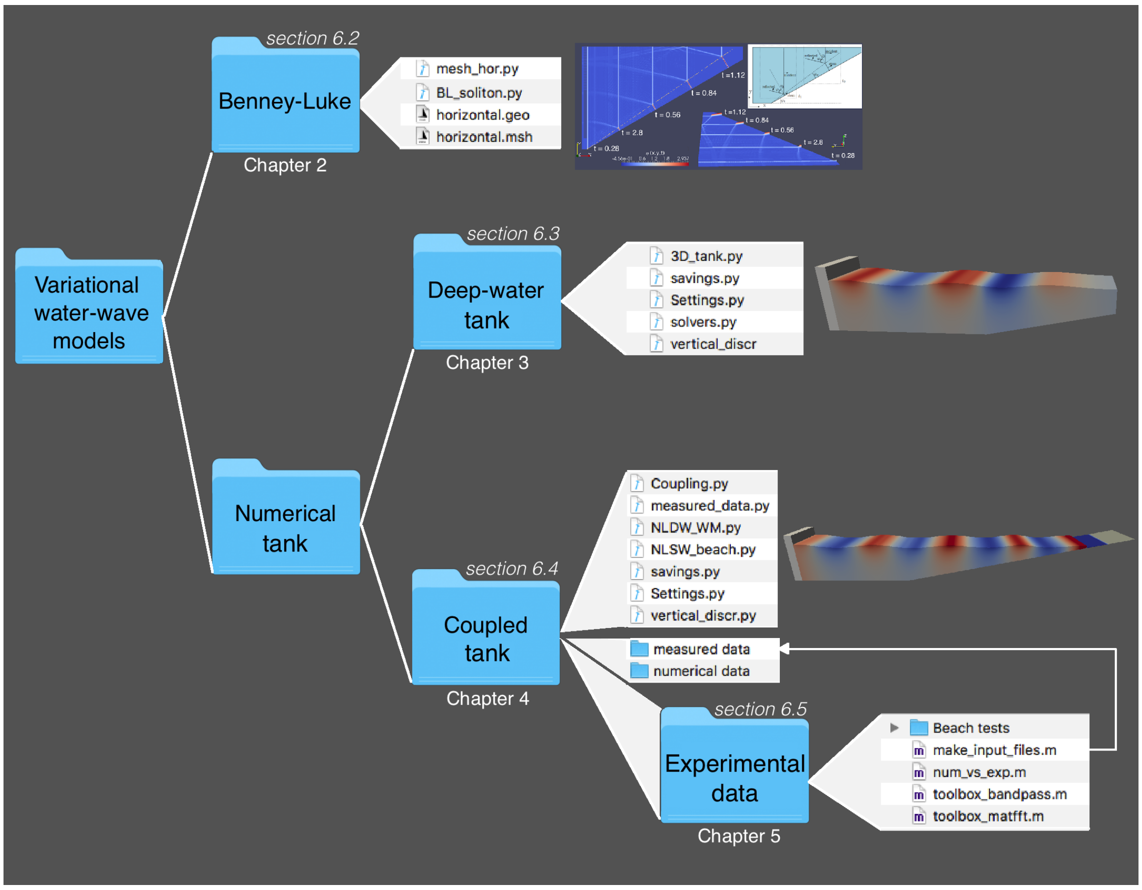


Figure 6.1: Code sources

the solitary-wave simulations. You can download it on `http://gmsh.info/#Download`. Finally, Paraview [3] is used for visualisation of the solutions; it can be downloaded here: `https://www.paraview.org/download/`.

6.2 Rogue-type waves in shallow water: Benney-Luke model for oblique interaction of solitary waves

6.2.1 Introduction

In this code, we simulate extreme waves that occur due to Mach reflection through the intersection of a solitary wave with an oblique wall. For a given range of incident angles and amplitudes, the Mach stem wave grows linearly in length and amplitude, potentially reaching up to four times the amplitude of the incident waves. A variational approach was used in Chapter 2 to derive the bidirectional Benney–Luke equations, an asymptotic equivalent of the three-dimensional potential-flow equations modelling water waves. This nonlinear and weakly dispersive model has the advantage of allowing wave propagation in two horizontal directions, which is not the case with the unidirectional Kadomtsev–Petviashvili (KP) equation used in most previous studies. A variational Galerkin finite-element method is applied to solve the system numerically in Firedrake with a second-order Störmer–Verlet temporal integration scheme, in order to obtain stable simulations that conserve the overall mass and energy of the system. Using this approach, we are able to get close to the fourfold amplitude amplification predicted by Miles.

6.2.2 Code source

The codes are in the folder *Benney Luke*, that contains two python files.

- *mesh_hor.py* is used to define the domain characteristics, such as the length of the incident channel and oblique wall, and the angle of incidence. Running this file will create the *horizontal.geo* file, that can be opened in Gmsh.
- The main file *BL_soliton.py*, in which the solvers are implemented and solved.

Running the file *BL_soliton.py* creates the following solution files:

- *eta.pvd* is the free-surface solution; open it on Paraview and apply a "*Wrap by scalar*" filter to observe the free-surface evolution;
- *phi.pvd* is the velocity-potential solution. Open it on Paraview to observe its evolution;
- *Ampl.txt* saves the time (first column), the amplitude of the stem wave (second column) and the amplitude of the incident soliton (third column). Open it with, *e.g.* Matlab, to compare the evolution of the incident and stem waves.

6.2.3 Use of the code

One first needs to define the domain characteristics with the file *mesh_hor.py*. Running this file will create the *horizontal.geo* file, that can be opened in Gmsh. Then, one must create the 2D mesh *horizontal.msh* with Gmsh, by selecting $mesh \rightarrow 2D \rightarrow save$. Finally, one can run the main file *BL_soliton.py*, making sure that the length of the channel, the length of the oblique wall and the angle of incidence are the same as in the *mesh_hor.py* file. It is highly recommended to run this file in parallel to reduce the computational time.

6.2.4 Code description

A tutorial to solve the Benney-Luke equations with the Störmer-Verlet scheme, published by A. Kalogirou and O. Bokhove, is available in the Firedrake documentation : `https://www.firedrakeproject.org/demos/benney_luke.py.html`. The extended program used to perform the numerical simulations of Chapter 2 is archived on Zenodo (`https://zenodo.org/badge/latestdoi/79556994`) and detailed hereafter.
First, firedrake and numpy are imported at the top of file *benney_luke.py* through

```
from firedrake import *
import numpy as np
```

The parameters defining the domain and the initial soliton must then be defined. Details on how to choose these parameters are given in Chapter 2. As a remainder, the domain is defined by a horizontal channel of length L_c and width L_d, closed by the oblique wall of length L_w and angle ψ with the x–direction. The Benney-Luke equations are based on the small-amplitude and small-dispersion parameters, ϵ and μ, which must therefore be given. One must also set the initial time and final times t and T_{end} respectively, and the time step Δt.

```
"""
**************************************************
*                   Parameters                   *
**************************************************"""
Lw = 500.0                      # Length of the wall
Lc = 5.0            # Length of the incident channel
psi = pi/6               # Angle of the oblique wall
Ld = Lw*sin(psi)     # Width of the incident channel
ep = 0.19                # Small amplitude parameter
mu = 0.02               # Small dispersion parameter
psi_inc = 0.0          # angle of the initial soliton
dx = 0.4              # x-refinement in largest areas
dy = 1.5              # y-refinement in largest areas
dxx = 0.25             # x-refinement in finest areas
dyy = 0.25             # y-refinement in finest areas
```

```
t=0.0                                 # Initial time
t_save = t                             # Saving time
dt = 0.0028                              # Time step
Tend = 150.0                            # Final time
```

These parameters must be used to generate the mesh in *mesh_hor.py* as well. Once the mesh file *horizontal.msh* is created (*cf.* how to create it in section 6.2.3), it is loaded in the main file through:

```
"""
****************************************************
*                       Mesh                       *
****************************************************"""
mesh = Mesh("horizontal.msh")      # Load the mesh file
coords = mesh.coordinates       # access to coordinates
```

The function space is defined on the mesh, here based on second-order continuous Galerkin expansions as defined in line 6 of the code below. Then, the solutions η, ϕ and q at times t^n (denoted by *_n0*), $t^{n+1/2}$ (denoted by *_half*) and t^{n+1} (denoted by *_n1*) are defined on this function space. The linear solvers are solved for the unknowns η, ϕ and q that are introduced through trial functions. Finally, the basis function is represented by v through the test function of Firedrake.

```
1  """
2  *********************************************
3  *                 Functions                 *
4  *********************************************"""
5  # -------------- Function Space -------------- #
6  V = FunctionSpace(mesh, "CG", 2)  # Vector space
7  
8  # ----------- Define the functions ----------- #
9  eta_n0 = Function(V)                    # eta(n)
10 phi_n0 = Function(V)                    # phi(n)
11 q_n0 = Function(V)                        # q(n)
12 eta_n1 = Function(V)                  # eta(n+1)
13 phi_n1 = Function(V)                  # phi(n+1)
14 q_n1 = Function(V)                      # q(n+1)
15 eta_half = Function(V)              # eta(n+1/2)
16 phi_half = Function(V)              # phi(n+1/2)
17 q_half = Function(V)                  # q(n+1/2)
```

```

# ----------------- Unknowns ---------------- #
eta = TrialFunction(V)
phi = TrialFunction(V)
q = TrialFunction(V)

# -------------- Test function -------------- #
v = TestFunction(V)
```

The solutions are initialised by interpolating the expression of the exact soliton solutions (2.37) derived in Chapter 2.

```
"""
*******************************************************************************
*                              Initial solution                              *
*******************************************************************************"""
# --------------------------- Soliton's parameters ---------------------------- #
A = 1.0                                                             # Amplitude
C=0.5*(A + tan(psi)*tan(psi)/ep)                                   # Constant C
dist = 0.5                                # distance(%) from the channel boundary
x0 = dist*Lc                                   # initial soliton position (x0,y0)
y0 = 0.0

# ------------------------- Expression of eta and phi ------------------------- #
expr_eta = Expression("A*pow(cosh(sqrt(3*ep*A/(4*mu))*((x[0]-x0)  \
                       + (x[1]-y0)*tan(psi)-(t-t0)*(1+C*ep))),-2)", A=A, x0=x0,\
                        y0=y0, psi=psi_inc, ep=ep, mu=mu, C=C, t=t, t0=0.0)

expr_phi = Expression("A*sqrt(4*mu/(3*ep*A))*(tanh(sqrt(3*ep*A/(4*mu))*((x[0]-x0)\
                       + (x[1]-y0)*tan(psi)-(t-t0)*(1+C*ep)))+1) ", A=A, x0=x0, \
                       y0=y0, psi=psi_inc, ep=ep, mu=mu, C=C, t=t, t0=0.0)

# ------------------------------- Initialization ------------------------------- #
phi_n0.interpolate(expr_phi)
eta_n0.interpolate(expr_eta)
```

In Firedrake, the weak formulations are defined in their space-continuous-time-discrete form. Variational problems and solvers may be called directly as follows.

```
"""
***************************************************************************
*                              Weak formulations                           *
***************************************************************************"""
# --------------------------- Update phi(n+1/2) --------------------------- #
F_phi_half = (v*(phi_half - phi_n0)/(dt/2.0) + v*eta_n0 \
             + 0.5*mu*inner(grad(v),grad((phi_half-phi_n0)/(dt/2.0)))\
             + 0.5*ep*v*inner(grad(phi_half),grad(phi_half)))*dx
phi_problem_half = NonlinearVariationalProblem(F_phi_half,phi_half)
phi_solver_half = NonlinearVariationalSolver(phi_problem_half)

# ---------------------------- Update q(n+1/2) ---------------------------- #
a_q_half = v*q*dx
L_q_half = 2.0/3.0*inner(grad(v),grad(phi_half))*dx
q_problem_half = LinearVariationalProblem(a_q_half,L_q_half,q_half)
q_solver_half  = LinearVariationalSolver(q_problem_half)

# ---------------------------- Update eta(n+1) ---------------------------- #
a_eta = (v*eta/dt + 0.5*mu*inner(grad(v),grad(eta/dt)) \
        - 0.5*inner(grad(v),grad(phi_half))*ep*eta)*dx
L_eta = (v*eta_n0/dt + 0.5*mu*inner(grad(v),grad(eta_n0)/dt) \
        +  mu*inner(grad(v),grad(q_half))\
        + 0.5*inner(grad(v),grad(phi_half))*(2+ep*eta_n0))*dx
eta_problem = LinearVariationalProblem(a_eta,L_eta, eta_n1)
eta_solver  = LinearVariationalSolver(eta_problem)

# ---------------------------- Update phi(n+1) ---------------------------- #
a_phi_n1 = (v*phi/(dt/2) + 0.5*mu*inner(grad(v),grad(phi/(dt/2))))*dx
L_phi_n1 = (v*phi_half/(dt/2) + 0.5*mu*inner(grad(v),grad(phi_half)/(dt/2))\
           - v*eta_n1 - 0.5*ep*v*inner(grad(phi_half),grad(phi_half)))*dx
phi_problem_n1 = LinearVariationalProblem(a_phi_n1,L_phi_n1, phi_n1)
phi_solver_n1  = LinearVariationalSolver(phi_problem_n1)
```

The solvers are then called in the time loop to update the solutions:

```
"""
********************************************************************************
*                                    Time loop                                 *
********************************************************************************"""
while(t < Tend):
    # -------------------------------- Save data -------------------------------- #
    if t >= t_save:
        with eta_n0.dat.vec_ro as eta_v, inf_y.dat.vec_ro as inf_v:
            _, eta_max = eta_v.max()                        # stem wave's amplitude
            tmp = inf_v.duplicate()
            tmp.pointwiseMult(eta_v, inf_v)
            _, init_max = tmp.max()                     # incident wave's amplitude
        Ampl_file.write('%-10s %-10s %-10s \n' % (t, eta_max,init_max))
        eta_file.write(eta_n0)                # Save the surface deviation solution
        phi_file.write(phi_n0)                   # Save the potential flow solution
        t_save = t_save + dt_save                         # Update the saving time
        print(t/Tend)                                          # Print progression

    # ------------------------------- Update time ------------------------------ #
    t += dt

    # ----------------------- Solve the weak formulations ---------------------- #
    phi_solver_half.solve()                                       # Get phi^{n+1/2}
    q_solver_half.solve()                                           # Get q^{n+1/2}
    eta_solver.solve()                                              # Get eta^{n+1}
    phi_solver_n1.solve()                                           # Get phi^{n+1}

    # -------------------------- Update the solutions -------------------------- #
    phi_n0.assign(phi_n1)
    eta_n0.assign(eta_n1)
```

In the above code, the first part saves the stem wave's amplitude, that is, the maximal free-surface deviation, and the incident wave's amplitude by looking for the maximal amplitude near the upper y–boundary, $y = L_d$, based on the assumption that despite growing in length, the stem wave sticks to the oblique wall and does not reach the upper y boundary (*cf.* the location of the stem and incident waves in Fig. 6.1). The Heaviside function inf_y is therefore equal to 1 in the domain $\{0 \leq x \leq L_x; L_d - 4.0m \leq y \leq L_y\}$ and equal to 0 elsewhere:

```
inf_y=Function(V).interpolate(Expression("0.5*(1+copysign(1.0,x[1]-(Ld-4.0))),Ld=Ld"))
```

The saving time *t_save* is previously initialised to zero and enables to save the solutions at the time interval dt_save. The solutions η and ϕ in the full domain are also saved and may be observed on Paraview. The corresponding files were created before entering the time loop through:

```
"""
******************************************************************************
*                                Saving files                                *
******************************************************************************"""
phi_file = File('data/phi.pvd')                # potential phi numerical solution
eta_file = File('data/eta.pvd')         # surface deviation eta numerical solution
Ampl_file = open('data/amplitudes.txt', 'w') # Incident and stem waves' amplitudes
```

The next part of the code, at line 17, updates time, meaning that we now consider $t = t^{n+1}$. The solvers are called to update the functions with the second-order Störmer-Verlet scheme. Finally, at the end of the time loop, the solutions are updated for the next time step.

6.3 Three-dimensional wave tank with wavemaker and seabed topography

6.3.1 Introduction

Firedrake has been used to solve nonlinear potential-flow equations in a deep-water domain with seabed topography. Waves are generated by a piston wavemaker on the left-hand side of the basin and reflected on a vertical wall on the right-hand side of the basin. Spatial discretisation strategies were derived in Chapter 3 to deal with moving boundaries at the wavemaker and at the free surface, as well as to update the vertical structure of the velocity potential. A variational approach has been used to derive the equations, leading to Hamiltonian dynamics on which both a first-order (symplectic-Euler) and a second-order (Stormer-Verlet) energy-conserving time schemes have been applied, respectively, to ensure stability. This section explains to the user which parameters must be specified and details the code to ease future extensions.

6.3.2 Code sources

The code consists four main files.

- The file *Settings.py* in which you need to specify the domain, time and wavemaker settings;
- The file $vertical\_discr.py$, in which the vertical discretisation is implemented. It includes the computation of the Lagrange expansions and vertical matrix coefficients;

- The file $solvers.py$, which contains the weak formulations;
- The file $savings.py$, which contains the functions used to save the data;
- The file $3D\_tank.py$, in which the solvers are defined and called to update the solutions.

To observe the free-surface and velocity-potential solutions, open the file *waves.pvd* on Paraview, which contains the value of the velocity potential $\phi(x, y, t)$. The free-surface and the left boundary of the domain move as $h(x, y, t) = H(x) + \eta(x, y, t)$ and $R(y, t)$ respectively. The wavemaker motion is also saved both as a text file *wm_motion.txt* and in a function *wavemaker.pvd* that can be observed in Paraview after applying the filter "*Wrap by scalar*". The energy is also saved in the file *energy.txt*. Finally, a *README.txt* file is created at the end of the simulations to sum up the parameters used to create the data.

6.3.3 Use of the code

In order to run the codes, you first need to set the beach, wavemaker, domain and time parameters in the file *Settings.py*.

First, choose whether you create the wavemaker signal or import it from measurements. In this tutorial, only the case with a created wavemaker signal is considered; details on how to import measurement data are given in section 6.5. The dimension (2D or 3D) and temporal scheme (symplectic Euler or Störmer-Verlet) must also be specified. A path to save the data is created automatically, but may be changed in the *test_case()* function.

```
"""
    *********************************************
    *                  Test case                *
    *********************************************"""
def test_case():
    #________________ Numerical Tank _______________#
    tank = "DeepWater"           # PF eq. in DW tank
    #tank = "Coupled"     # Coupling PF with SW beach
    #_________________ Kind of data ________________#
    #input_data = "measurements"  # from experiments
    input_data = "created"        # set the wavemaker
    #_______________ Temporal scheme _______________#
    scheme = "SE"
    #  "SE": Symplectic-Euler ; "SV": Stormer-Verlet
    #__________________ Dimension __________________#
    dim = "2D"
    #"2D": R(t) and b(x); "3D": R(y,t) and/or b(x,y)
    # if input = measurements, the dim must be 2D.
    #______ Path and name of the saved files ______#
    save_path = tank +'/data/' +scheme+'/'+ dim +'/'
    return tank, input_data, scheme, dim, save_path
```

The dimensions of the numerical domain, including the beach parameters, are defined in the function *domain()*. In this example, the seabed topography is assumed to take the form

$$H(x) = H_0 - b(x), \quad \text{with} \begin{cases} b(x \leq x_b) = 0 \\ b(x > x_b) = s_b x. \end{cases} \tag{6.6}$$

The characteristics of the seabed topography thus include the maximal depth at rest H_0, the coordinate at which the beach starts x_b and the slope $s_b > 0$. The characteristics of the numerical domain include the depth at the end of the domain, H_{end}, the length in the x–direction, L_x, the length in the y–direction, L_y, the length on which to apply the x–transform, L_w, the resolutions in x and y, res_x and res_y respectively, and the number of vertical layers n_z. To avoid instability of the numerical simulations, one must ensure non-negative depth at the end of the domain. Moreover, our discretisation of the potential-flow model does not hold discontinuous waves such as breaking waves, so the depth at the end of the domain must satisfy: $H_{end} \gg \lambda/20$, with λ the wavelength, to avoid steep or breaking waves and ensure stability of the simulations. One way to ensure stability is therefore to prescribe $H_{end} \gg \lambda/20$ and deduce the length L_x from the seabed topography:

$$L_x = x_b + \frac{H_0 - H_{end}}{s_b}. \tag{6.7}$$

To ensure accuracy of the simulations, the spatial resolution is set to satisfy $\Delta x < \lambda/10$.

```
"""
    ************************************************************************
    *                          Numerical domain                            *
    ************************************************************************"""
def domain():
    #_______________________ Beach _______________________#
    H0 = 1.0                            # Depth at rest (flat bottom)
    xb = 4.0                                      # Start of the beach
    sb = 0.2                                      # Slope of the beach
```

```
10     H_expr = Expression("H0-0.5*(1+copysign(1.0,x[0]-xb))*slope*(x[0]-xb)",
11                         H0=H0,xb=xb, slope=sb)
12 
13     #________________________ Basin _______________________#
14     Hend = 0.5                         # Depth at the end of the beach
15     Lx = xb +(H0-Hend)/sb                                # Length in x
16     Ly = 1.0                                             # Length in y
17     Lw = 1.0                                  # End of the x-transform
18     res_x = 0.05                                        # x-resolution
19     res_y = 0.2                                         # y-resolution
20     n_z = 8                                   # Order of the expansion
21     return H0, xb, sb, H_expr, Hend, Lx, Ly, Lw, res_x, res_y, n_z
```

The function “copysign” used at line 10 is used to apply the Heaviside function resulting in (6.6). In the example given here, the wavemaker motion is defined as

$$R(y,t) = \gamma \hat{R}(y) \cos(\omega t), \tag{6.8}$$

with

$$\hat{R}(y) = \begin{cases} \dfrac{2y - L_y}{Ly} & \text{if dim = 3D,} \\ 1, & \text{if dim = 2D.} \end{cases} \tag{6.9}$$

The characteristics of the wavemaker thus include its frequency ω, its period T_w, and its amplitude γ. Note that ω and T_w are chosen with respect to the wavelength λ of the waves. We also define a variable t_stop in order to stop the wavemaker after some time. To limit extra frequencies due to an abrupt stop of the wavemaker motion, we set $t\_stop = N * T_w$, where $N \in \mathbb{N}$, so that at $t = t\_stop$ the velocity of the wavemaker is null. The wavemaker characteristics, expression and derivatives can be changed through the function *wavemaker()*:

```
"""
    ***************************************************************************
    *                                Wavemaker                               *
    ***************************************************************************"""
def wavemaker(dim, H0, Ly, Lw, t):
    #______________________________ Characteristics ______________________________#
    g = 9.81                                           # Gravitational constant
    lamb = 2.0                                                     # Wavelength
    k = 2*pi/lamb                                                 # Wave number
```

```
10      w = sqrt(g*k*tanh(k*H0))                                 # Wave frequency
11      Tw = 2*pi/w                                                 # Wave period
12      gamma = 0.03                                             # Wave amplitude
13      t_stop = 5.0*Tw                              # When to stop the wavemaker
14  
15      #________________________________ Expression _______________________________#
16      if dim == "2D":
17          WM_expr = \
18          Expression("-0.5*(1+copysign(1.0,Lw-x[0]))*A*cos(w*t)",A=gamma, Lw = Lw, w=w, t=t)
19      elif dim == "3D":
20          WM_expr = \
21          Expression("-0.5*(1+copysign(1.0,Lw-x[0]))*A*(x[1]-0.5*Ly)/(0.5*Ly)*cos(w*t)", A=gamma, Ly=Ly, Lw = Lw, w=w, t=t)
22  
23      #_____________________________ Time derivative _____________________________#
24      if dim == "2D":
25          dWM_dt_expr = \
26          Expression("0.5*(1+copysign(1.0,Lw-x[0]))*A*w*sin(w*t)",A=gamma, Lw=Lw, w=w, t=t)
27      elif dim == "3D":
28          dWM_dt_expr = \
29          Expression("0.5*(1+copysign(1.0,Lw-x[0]))*A*w*(x[1]-0.5*Ly)/(0.5*Ly)*sin(w*t)",A=gamma, Ly=Ly, Lw=Lw, w=w, t=t)
30      #______________________________ y-derivative _______________________________#
31      if dim == "2D":
32          dWM_dy_expr = Expression("0.0")
33      elif dim == "3D":
34          dWM_dy_expr = Expression("-0.5*(1+copysign(1.0,Lw-x[0]))*A*cos(w*t)/(0.5*Ly)", A=gamma, Ly=Ly, Lw=Lw, w=w, t=t)
35  
36      return g, lamb, k, w, Tw, gamma, t_stop, WM_expr, dWM_dt_expr, dWM_dy_expr
```

Note that the function *copysign* in lines 18 and 21 is used to apply the Heaviside function so that R and its derivatives are null for $x > L_w$ (*cf.* Chap.3 for more details).

Finally, define the initial and final times as well as the time step and the time interval *dt_save* after which the data are saved with the function *set_time()* that depends on the wavemaker period "WM_period":

```
"""
    ***********************************
    *               Time              *
    ***********************************"""
def set_time(WM_period):
    T0 = 0.0                # Initial time
    Tend = 10*WM_period       # Final time
    t = T0             # Temporal variable
    dt = 0.001                  # Time-step
    dt_save = 0.02      # Saving time step
    return T0, t, dt, Tend, dt_save
```

The main code *3D_tank.py* can then be run to obtain the simulations. The above settings are sufficient to adapt the code to your needs.

6.3.4 Code description

The main file *3D_tank.py*, the vertical discretisation *vertical_discr.py* and the saving strategies *savings.py* are detailed next to allow further extension.

Import the settings

In the main file, the libraries and functions are imported through

```
import numpy as np
import os.path
from vertical_discr import *
from savings import *
from Settings import *
from firedrake import *
import solvers as DW_solvers
```

Then, the settings are applied through a call to the above mentioned functions:

```
"""
    **********************************************************
    *                        Settings                        *
    ********************************************************** """
input_data, scheme, dim, save_path = test_case()
H0, xb, sb, H_expr, Hend, Lx, Ly, Lw, res_x, res_y, n_z = domain()
T0, t, dt, Tend, dt_save = set_time()
```

```
g, lamb, k, w, Tw, gamma, t_stop, WM_expr,
    dWM_dt_expr, dWM_dy_expr = wavemaker(dim, H0, Ly, Lw, t)
```

Definition of the numerical functions

As explained in Chapter 3, our strategy is to solve the weak formulations in each horizontal layer, meaning that the solutions h, ψ_1 and $\psi_{i'}$ are z–independent and only defined in the horizontal plane. In the case of pluri-directional waves (3D), the mesh is defined as a rectangular surface with quadrilateral elements. However, when the wavemaker is constant in the y-direction, the generated waves will only be x– and z–dependent and the domain may be simplified to a 2D vertical tank to reduce the computational time. The horizontal solutions are then only dependent on the x–coordinate, on a 1D mesh:

```
"""
    ****************************************************
    *               Definition of the mesh             *
    **************************************************** """

#_________________ Vertical discretization ________________#
Nz = n_z+1                  # Number of point in one element

#________________ Horizontal discretization _______________#
Nx = round(Lx/res_x)               # Number of elements in x
Ny = round(Ly/res_y)               # Number of elements in y

#__________________________ Mesh __________________________#
if dim=="2D":                                    #(x,z)-waves
    hor_mesh = IntervalMesh(Nx,Lx)
else:                                          #(x,y,z)-waves
    hor_mesh = RectangleMesh(Nx,Ny,Lx,Ly,quadrilateral=True)
```

The function spaces of the variables are then defined from the mesh. The depth h and the free-surface velocity potential ψ_1 are expanded as C^0 continuous Galerkin functions, for example as first-order polynomials. The interior velocity potential $\hat{\psi}$ is a vector of n_z components, each corresponding to the expansion of the velocity potential on the corresponding horizontal layer. We therefore introduce a specific function space for the interior velocity, through a vector of dimension n_z for which each component is expanded as C^0 continuous Galerkin functions, *i.e.* as 1^{st}–order polynomials. As the equations for h and $\hat{\psi}$ are solved in unison, we also introduce a mixed function space (that is, a function space for coupled variables that are initially defined on different function spaces) that combines the space of definition of h and the one of $\hat{\psi}$.

```
"""
    ************************************************
    *       Definition of the function spaces      *
    ************************************************ """
#___________________ For h and psi_1 ___________________#
V = FunctionSpace(hor_mesh, "CG", 1)
#_____________________ For hat_psi _____________________#
Vec = VectorFunctionSpace(hor_mesh, "CG", 1, dim=n_z)
#_________________ Mixed function space ________________#
V_mixed = V*Vec # to solve simultaneous weak formulations
```

The basis functions are defined on each function space as follows

```
#______________________ Test functions ______________________#
delta_h = TestFunction(V)                         # from dH/dh
delta_hat_psi = TestFunction(Vec)           # from dH/dhat_psi
w_t = TestFunction(V_mixed)                # from dH/dpsi_1...
delta_psi, delta_hat_star = split(w_t)    # ...and dH/dhat_psi
```

The functions involved in the symplectic-Euler and Störmer-Verlet temporal schemes are defined on their respective space of definition, with index $n0$ at time t^n, index $star$ at auxiliary time t^*, index $half$ at time $t^{n+1/2}$ and index $n1$ at time t^{n+1}:

```
"""
    ******************************************************
    *            Definition of the functions             *
    ****************************************************** """
if scheme=="SE": #_________ Symplectic-Euler scheme _________#
    #_____________________ At time t^n _____________________#
    h_n0 = Function(V)                                  # h^n
    psi_1_n0 = Function(V)                          # psi_1^n
    hat_psi_n0 = Function(Vec)                    # hat_psi^n

    #_______________ At time t^{n+1} and t^* _______________#
    psi_1_n1 = Function(V)                      # psi_1^{n+1}
    w_n1 = Function(V_mixed)
    h_n1, hat_psi_star = split(w_n1)      # h^{n+1}, hat_psi^*
    hat_psi_n1 = Function(Vec)     # to visualise hat_psi^{n+1}
```

```
else: #________________ Stormer-Verlet scheme _______________#
    #______________________ At time t^n _____________________#
    h_n0 = Function(V)                                   # h^n
    psi_1_n0 = Function(V)                           # psi_1^n
    hat_psi_n0 = Function(Vec)                     # hat_psi^n

    #_______________ At time t^{n+1/2} and t^* ______________#
    w_half = Function(V_mixed)        # to obtain psi^{n+1/2},
    psi_1_half, hat_psi_star = split(w_half)   # and hat_psi^*

    #_______________ At time t^{n+1} and t^** _______________#
    psi_1_n1 = Function(V)                       # psi_1^{n+1}
    w_n1 = Function(V_mixed)              # to obtain h^{n+1},
    h_n1, hat_psi_aux = split(w_n1)         # and hat_psi^{**}
    hat_psi_n1 = Function(Vec)    # to visualise hat_psi^{n+1}
```

Some of the equations are explicit updates of ψ_1 or $\hat{\psi}$, and may thus be solved with a linear solver in order to reduce the computational time. We thus introduce trial functions for which the temporarily-linear weak formulations will be solved:

```
#______________________ Trial functions _____________________#
psi_1 = TrialFunction(V)      # psi_1^{n+1} for linear solvers
hat_psi = TrialFunction(Vec)# hat_psi^{n+1} for linear solvers
```

The beach topography $b(x)$, the depth at rest $H(x)$ and the wavemaker function $\tilde{R}(x, y, t)$ are also discretised on the function space V as follows:

```
#___________________________ Beach __________________________#
beach = Function(V)                                     # b(x)
#_______________________ Depth at rest ______________________#
H = Function(V)                                         # H(x)
#_________________________ Wavemaker ________________________#
WM = Function(V)                                  # R(x,y;t^n)
dWM_dt = Function(V)                               #  (dR/dt)^n
dWM_dy = Function(V)                               #  (dR/dy)^n
if scheme=="SV":                         # For Stormer-Verlet:
    WM_half = Function(V)                   # R(x,y;t^{n+1/2})
    dWM_half_dt = Function(V)                #  (dR/dt)^{n+1/2}
    dWM_half_dy = Function(V)                #  (dR/dy)^{n+1/2}
WM_n1 = Function(V)                           # R(x,y;t^{n+1})
dWM_n1_dt = Function(V)                        #  (dR/dt)^{n+1}
dWM_n1_dy = Function(V)                        #  (dR/dy)^{n+1}
```

```
#________________________ x coordinate _______________________#
x_coord = Function(V).interpolate(Expression('x[0]'))
```

The last line defines the x–coordinate as a function as it is explicitly used in the weak formulations. Now that the functions are defined in their respective function spaces, the prescribed functions are initialised by interpolation of the corresponding expressions:

```
"""
   *********************************************
   * Initialisation of the prescribed functions *
   *********************************************"""
#____________________ Topography ___________________#
H.interpolate(H_expr)             # Depth at rest H(x)
beach.interpolate(H0-H)                   # Beach b(x)
#____________________ Wavemaker ___________________#
WM.interpolate(WM_expr)                  # \tilde{R}
dWM_dy.interpolate(dWM_dy_expr)              # dR/dy
dWM_dt.interpolate(dWM_dt_expr)              # dR/dt
```

To initialise the depth and velocity potential, we consider that at the initial time $t = T_0$, the fluid is at rest, meaning that $h = H(x)$ and $\phi(x, y, t) = 0.0$. We thus initialise the solutions as follows:

```
#_________________ Depth h(x,y,t) _________________#
h_n0.assign(H)                   # h(x,y;t=0) = H(x)
w_n1.sub(0).assign(H)   # First estimate for h^{n+1}

#______ Surface velocity pot. phi(x,y,z=h,t) ______#
psi_1_n0.assign(0.0)           # \psi_1(x,y;t=0) = 0
if scheme =="SV":
    w_half.sub(0).assign(0.0)       # \psi_1^{n+1/2}

#_____ Vel. pot. in depth: phi(x,y,z<h,t) = 0 _____#
for i in range(0,n_z):
    hat_psi_n0.dat.data[:,i] = 0.0          # psi_i^n
    w_n1.sub(1).dat.data[:,i] = 0.0      # psi_i^{*}
    if scheme=="SV":
        w_half.sub(1).dat.data[:,i]= 0.0 # psi_i^{*}
```

Vertical discretisation

While the horizontal discretisation is made internally with Firedrake, the vertical matrices must be evaluated (semi-)analytically in order to be used as coefficients in the horizontal variational principle and the resulting weak formulations. This is done in the file *vertical_discr.py*.

As explained in Chapter 3, the domain in the z–direction is effectively and implicitly discretised with one element on which the vertical component $\tilde{\varphi}_i(z)$ of the velocity potential is expanded through polynomials of order n_z. In this example, Lagrange polynomials are used:

$$\tilde{\varphi}_i(z) = \prod_{\substack{k=1\\k\neq i}}^{n_z+1} \frac{z - z_k}{z_i - z_k}. \tag{6.10}$$

We use the Python library for symbolic mathematics ”Sympy” to evaluate these matrices semi-analytically.

```python
from sympy import *
```

First, we define a function *varphi_expr()* that returns the polynomial $\tilde{\varphi}_i$ as an expression of the coordinate z. This function requires the index i of $\tilde{\varphi}$, the order n_z of the polynomial, and the upper limit of the domain, here given by H_0:

```python
""" ***************************************************************
    *                       Lagrange polynomial                   *
    ***************************************************************
    This function gives the expression of the Lagrange polynomial
    varphi_i(z) of order n, for z between z=0 and z=H_0.         """

def varphi_expr(i, n, H0):
    z=Symbol('z')                                      # z-coordinate
    k = Symbol('k')                           # index k in the product
    z_k = H0*((n-k)/n)                      # discrete coordinates z_k
    sigma = lambdify(k,z_k,"numpy")                   # sigma(k) = z_k
    varphi_z = \
    (Product((z-sigma(k))/(sigma(i)-sigma(k)),(k,0,i-1))\
     *Product((z-sigma(k))/(sigma(i)-sigma(k)),(k,i+1,n))).doit()
    return varphi_z
```

In this code, we introduce the coordinate z as a symbol so that the polynomial can be evaluated in terms of z. Similarly, we introduce the symbol k that will be used as the index of the discrete coordinates z_k. In this example, the evaluations of ϕ in the vertical are linearly distributed from $z = H_0$ to $z = 0$, so we define z_k as

$$z_k = H_0 \frac{n-k}{n}, \quad \text{for } k = 0, 1, \dots n. \tag{6.11}$$

However, if required, one may change the definition of z_k to obtain non-homogeneous (such as exponential) evaluations of ϕ over the depth. The variable *sigma* is created such that *sigma(k)* returns the discrete coordinate z_k. This function is called when computing the Lagrange polynomial *varphi_z*. Note that the product from $k = 0$ to $k = n$ is split into a first product from $k = 0$ to $k = i - 1$ and another one from $k = i + 1$ to $k = n$, to exclude the case $k = i$ as this would lead to zero (cf. the above definition of the Lagrange polynomial).
Similarly, we introduce the function *deriv_varphi_expr()* that returns the analytical expression of the derivative of $\tilde{\varphi}_i$ with respect to z, that is, $d_z\tilde{\varphi}_i$:

```python
""" ************************************
    *           d(\varphi)/dz          *
    ************************************
    This function returns the expression
    of the z--derivative of the Lagrange
    polynomial, that is d(\varphi)/dz . """

def deriv_varphi_expr(varphi_expr):
    z = Symbol('z')          # coordinate z
    deriv = diff(varphi_expr,z)
    return deriv         # d(\varphi(z))/dz
```

From the two above functions, we are able to compute each of the matrices (3.20) associated with the vertical basis functions through functions of the form $X_{ij}(i, j, n, H0)$ that return the (i, j) component of matrix X, in terms of $\tilde{\varphi}_i(z)$ and $\tilde{\varphi}_j(z)$:

```python
""" ***********************************************
    *                Vertical matrices            *
    *********************************************** """
# Mass matrix
def M_ij(i,j,n,H0):
    z=Symbol('z')
    expr_M = varphi_expr(i,n,H0)*varphi_expr(j,n,H0)
    M = integrate(expr_M, (z,0,H0))
    return M

# Laplace matrix
def A_ij(i,j,n,H0):
    z=Symbol('z')
    expr_A = deriv_varphi_expr(varphi_expr(i,n,H0))\
        *deriv_varphi_expr(varphi_expr(j,n,H0))
    A = integrate(expr_A, (z,0,H0))
    return A

def D_ij(i,j,n,H0):
    z=Symbol('z')
    expr_D = z*varphi_expr(i,n,H0)\
        *deriv_varphi_expr(varphi_expr(j,n,H0))
    D = integrate(expr_D, (z,0,H0))
    return D

def S_ij(i,j,n,H0):
    z=Symbol('z')
    expr_S = z*z*deriv_varphi_expr(varphi_expr(i,n,H0))\
        *deriv_varphi_expr(varphi_expr(j,n,H0))
    S = integrate(expr_S, (z,0,H0))
    return S

def I_i(i,n,H0):
    z=Symbol('z')
    expr_I = varphi_expr(i,n,H0)
    I = integrate(expr_I, (z,0,H0))
    return I
```

Each function is constructed the same way: first, we define the symbolic z–coordinate; then, we define the expression to integrate by calling the functions *varphi_expr()* and *deriv_varphi_expr()*; finally, we integrate this expression between $z = 0$ and $z = H_0$ with the in-build Sympy function $integrate$ that leads to the exact value of the matrix $\tilde{X}$ at indices (i, j).

In the main file, we then initialise and fill the matrices $\tilde{M}$, $\tilde{A}$, $\tilde{S}$, $\tilde{D}$ and $\tilde{I}$ (note that in the code we have omitted the tilde subscripts) and the corresponding submatrices used to distinguish the surface (index 1) and interior (subscript N) nodes by calling the above functions:

```
""" ************************
    * Compute the matrices *
    ************************ """
#_______ Initialization ______#
A = np.eye(Nz,Nz)*0.0
M = np.eye(Nz,Nz)*0.0
D = np.eye(Nz,Nz)*0.0
S = np.eye(Nz,Nz)*0.0
Ik = np.eye(Nz,1)*0.0

#____ Filling the matrices ___#
for i in range(0,Nz):
    for j in range(0,Nz):
        A[i,j]=A_ij(i,j,n_z,H0)
        M[i,j]=M_ij(i,j,n_z,H0)
        D[i,j]=D_ij(i,j,n_z,H0)
        S[i,j]=S_ij(i,j,n_z,H0)
    Ik[i] = I_i(i,n_z,H0)

#________ Submatrices ________#
A11 = A[0,0]
A1N = as_tensor(A[0,1:])
AN1 = as_tensor(A[1:,0])
ANN = as_tensor(A[1:,1:])

M11 = M[0,0]
M1N = as_tensor(M[0,1:])
MN1 = as_tensor(M[1:,0])
MNN = as_tensor(M[1:,1:])

D11 = D[0,0]
D1N = as_tensor(D[0,1:])
DN1 = as_tensor(D[1:,0])
DNN = as_tensor(D[1:,1:])

S11 = S[0,0]
S1N = as_tensor(S[0,1:])
SN1 = as_tensor(S[1:,0])
SNN = as_tensor(S[1:,1:])

I1 = Ik[0,0]
IN=as_tensor(Ik[1:,0])
```

As explained in Chapter 3, an advantage of this vertical discretisation is that the above submatrices are constant in both time and space and may thus be used as coefficients in the weak formulations without requiring any update in time.

Weak formulations

For clarity's sake, we only include here the weak formulations for the case of 2D waves, solved with the symplectic-Euler scheme. As explained in Chapter 3, the first step is to update h and $\hat{\psi}$ by solving simultaneously the following

$$h^{n+1} = h^n - \frac{\partial H(h^{n+1}, \psi_1^n, \psi_{i'}^*, W^n)}{\partial \psi_1^n}, \tag{6.12a}$$

$$0 = \frac{\partial H(h^{n+1}, \psi_1^n, \psi_{i'}^*, W^n)}{\partial \psi_{i'}^*}. \tag{6.12b}$$

These weak formulations, defined in (B.33) and (B.34), are implemented in Firedrake form in the file *solvers.py* as :

```
#----------------------------------------------------------------------------#
#  Step 1 : Update h at time t^{n+1} and psi_i at time t^* simulataneously:  #
#----------------------------------------------------------------------------#
def WF_h_SE(dim, n_z, g, H, H0, Lw, WM, dWM_dy, dWM_dt, dt, delta_psi, delta_hat_star,
     h_n0, h_n1, x_coord, psi_1_n0, hat_psi_star, M11, M1N, MN1, MNN, D11, D1N, DN1,
    DNN, S11, SN1, SNN, A11, AN1, ANN, I1, IN):

    WF_h = (H0*delta_psi*(h_n1-h_n0)*(Lw-WM)/dt -((h_n1/(Lw-WM))*(Lw*Lw)*(psi_1_n0.dx
    (0)*M11 + dot(hat_psi_star.dx(0),MN1))*delta_psi.dx(0)-( (1/(Lw-WM))*(Lw*Lw)*h_n1.
    dx(0))*( delta_psi.dx(0)*(D11*psi_1_n0 + dot(D1N,hat_psi_star)) +delta_psi*(
    psi_1_n0.dx(0)*D11 + dot(hat_psi_star.dx(0),DN1)))+(1/h_n1)*((1/(Lw-WM))*(Lw*Lw)*(
    h_n1.dx(0)**2))*(psi_1_n0*S11 + dot(hat_psi_star,SN1))*delta_psi+((Lw-WM)*H0*H0/
    h_n1)*(psi_1_n0*A11 + dot(hat_psi_star,AN1))*delta_psi -delta_psi*H0*(x_coord-Lw)*
    dWM_dt*h_n1.dx(0)))*dx - (delta_psi*Lw*dWM_dt*h_n1*I1)*ds(1)

    WF_hat_psi_star= ((h_n1/(Lw-WM))*(Lw*Lw)*elem_mult(delta_hat_star.dx(0),(MN1*
    psi_1_n0.dx(0)+dot(MNN,hat_psi_star.dx(0))))-((Lw*Lw)*h_n1.dx(0)/(Lw-WM))*(
    elem_mult(delta_hat_star, (psi_1_n0.dx(0)*D1N+ dot(DNN.T,hat_psi_star.dx(0)))) +
    elem_mult(delta_hat_star.dx(0),(DN1*psi_1_n0+dot(DNN,hat_psi_star)))) +(1.0/h_n1)
    *((Lw*Lw)*(h_n1.dx(0)**2)/(Lw-WM))*elem_mult(delta_hat_star,(SN1*psi_1_n0+ dot(SNN
    ,hat_psi_star)))+ ((Lw-WM)*H0*H0/h_n1)*elem_mult(delta_hat_star,(AN1*psi_1_n0+dot(
    ANN,hat_psi_star))))

    WF_hat_BC = (Lw*dWM_dt*h_n1*elem_mult(delta_hat_star,IN))
    WF_h_psi = WF_h + sum((WF_hat_psi_star[ind])*dx for ind in range(0,n_z)) + sum((
    WF_hat_BC[ind])*ds(1) for ind in range(0,n_z))

    return WF_h_psi
```

Then the surface velocity potential is subsequently updated explicitly through

$$W^{n+1}\psi_1^{n+1} = W^n\psi_1^n + \frac{\partial H(h^{n+1}, \psi_1^n, \psi_{i'}^*, W^n)}{\partial h^{n+1}}. \tag{6.13}$$

This equation, defined in (B.35), is implemented through a linear Firedrake weak formulation, as

```
#------------------------------------------------------------------------------#
#                    Step 2 : Update psi_1 at time t^{n+1}:                    #
#______________________________________________________________________________#
def WF_psi_SE(dim, g, H, H0, Lw, WM, WM_n1, dWM_dy, dWM_dt, dt, x_coord, delta_h,
    psi_1, psi_1_n0, hat_psi_star, h_n1, M11, MN1, MNN, D11, D1N, DN1, DNN,S11, SN1,
    SNN, A11, AN1, ANN, I1, IN):

A_psi_s = (delta_h*(Lw-WM_n1)*psi_1)*dx

L_psi_s = -(1/H0)*(-H0*delta_h*(Lw-WM)*psi_1_n0 +dt*(delta_h*((Lw*Lw)/(2.0*(Lw-WM)))
    *((psi_1_n0.dx(0)**2)*M11+dot(hat_psi_star.dx(0), (2.0*MN1*psi_1_n0.dx(0)+dot(MNN,
    hat_psi_star.dx(0)))))-((1.0/(Lw-WM))*(Lw*Lw)*delta_h.dx(0))*( psi_1_n0.dx(0)*(D11
    *psi_1_n0 + dot(D1N,hat_psi_star)) +dot(hat_psi_star.dx(0), (DN1*psi_1_n0 + dot(
    DNN, hat_psi_star))))+(1.0/h_n1)*(delta_h.dx(0)*((1.0/(Lw-WM))*h_n1.dx(0)*(Lw*Lw))
    -(delta_h/h_n1)*( (Lw*Lw)*(h_n1.dx(0)**2)/(2.0*(Lw-WM))))*(psi_1_n0*psi_1_n0*S11 +
     2.0*dot(hat_psi_star,SN1)*psi_1_n0+dot(hat_psi_star,dot(SNN,hat_psi_star)))-(0.5*
    delta_h*(Lw-WM)*H0*H0/(h_n1**2))*(psi_1_n0*psi_1_n0*A11 + 2.0*dot(hat_psi_star,AN1
    )*psi_1_n0 + dot(hat_psi_star,dot(ANN,hat_psi_star)))+H0*g*(Lw-WM)*delta_h*(h_n1-H
    ) - H0*psi_1_n0*(x_coord-Lw)*dWM_dt*delta_h.dx(0)))*dx - dt*(Lw*dWM_dt*delta_h*(
    psi_1_n0*I1 + dot(hat_psi_star,IN))/H0)*ds(1)

return A_psi_s, L_psi_s
```

Similar functions are defined for the ”3D” case and the Störmer-Verlet scheme. They are called in the main file to define the appropriate weak formulations as follows:

```
""" ************************************************************************
    *                            Weak Formulations                         *
    ************************************************************************ """
if scheme=="SE": #_____________________ Symplectic-Euler ______________________#
    # Step 1 : Update h at time t^{n+1} and psi_i at time t^* simulataneously: #
    WF_h_psi = DW_solvers.WF_h_SE(dim, n_z, g, H, H0, Lw, WM, dWM_dy, dWM_dt, dt,
    delta_psi, delta_hat_star, h_n0, h_n1, x_coord, psi_1_n0, hat_psi_star, M11, M1N,
    MN1, MNN, D11, D1N, DN1, DNN, S11, SN1, SNN, A11, AN1, ANN, I1, IN)

    #----------------- Step 2 : Update psi_1 at time t^{n+1}: -----------------#
    A_psi_s, L_psi_s = DW_solvers.WF_psi_SE(dim, g, H, H0, Lw, WM, WM_n1, dWM_dy,
    dWM_dt, dt, x_coord, delta_h, psi_1, psi_1_n0, hat_psi_star, h_n1, M11, MN1, MNN,
    D11, D1N, DN1, DNN,S11, SN1, SNN, A11, AN1, ANN, I1, IN)

    #----------------- Step 3 : Update psi_i at time t^{n+1}: -----------------#
    A_hat, L_hat = DW_solvers.WF_hat_psi_SE(dim, H, H0, g, n_z, Lw, x_coord, WM,
    dWM_dt, dWM_dy, dt, delta_hat_psi, hat_psi, h_n0, psi_1_n0, M11, MN1, MNN, D11,
    D1N, DN1, DNN,S11, SN1, SNN, A11, AN1, ANN, I1, IN)

elif scheme=="SV":#______________________ Stormer-Verlet ______________________#
    #--------------- Step 1 : Update psi_1^{n+1/2} and psi_i^*: ---------------#
    WF_psi_star = DW_solvers.WF_psi_half_SV(dim, n_z, g, H, H0, Lw, x_coord, WM,
    WM_half, dWM_dy, dWM_dt, dWM_half_dy, dWM_half_dt, dt, delta_psi, delta_hat_star,
    psi_1_n0, psi_1_half, hat_psi_star, h_n0, M11, M1N, MN1, MNN, D11, D1N, DN1, DNN,
    S11, SN1, SNN, A11, AN1, ANN, I1, IN)

    #----- Step 2 : Update h^{n+1} and psi_i at time t^** simulataneously: ----#
    WF_h_psi = DW_solvers.WF_h_SV(dim, n_z, Lw, H0, g, dt, x_coord, WM, WM_half,
    dWM_half_dy, dWM_half_dt, delta_psi, delta_hat_star, h_n0, h_n1, psi_1_half,
    hat_psi_star, hat_psi_aux, M11, M1N, MN1, MNN, D11, D1N, DN1, DNN, S11, SN1, SNN,
    A11, AN1, ANN, I1, IN)

    #----------------- Step 3 : Update psi_1 at time t^{n+1}: -----------------#
    a_psi_1, L_psi_1 = DW_solvers.WF_psi_n1_SV(dim, H0, H, g, x_coord, delta_h, Lw,
    WM_n1, WM_half, dt, psi_1_half, psi_1, dWM_half_dt, dWM_half_dy, hat_psi_aux, h_n1
    , M11, M1N, MN1, MNN, D11, D1N, DN1, DNN, S11, SN1, SNN, A11, AN1, ANN, I1, IN )

    #----------------- Step 4 : Update psi_i at time t^{n+1}: -----------------#
    A_hat, L_hat = DW_solvers.WF_hat_psi_SV(dim, n_z, Lw, H0, H, WM, x_coord, dt,
    dWM_dt, dWM_dy, delta_hat_psi, hat_psi, h_n0, psi_1_n0, M11, M1N, MN1, MNN, D11,
    D1N, DN1, DNN, S11, SN1, SNN, A11, AN1, ANN, I1, IN )
```

Solvers

Firedrake solvers solve variational problems based on parameters specified by the user; the choice of these parameters has been explained in Chapter 3.

```
"""
    **********************************************************************
    *                          Define the solvers                        *
    **********************************************************************"""
#__________________________ Solvers parameters __________________________#
param_h={"ksp_converged_reason":True,"pc_type": "fieldsplit","pc_fieldsplit_type": "
    schur","pc_fieldsplit_schur_fact_type": "upper"}
param_psi={"ksp_converged_reason":True,'ksp_type': 'preonly', 'pc_type': 'lu'}
param_hat_psi={"ksp_converged_reason":True,'ksp_type': 'preonly', 'pc_type': 'lu'}
```

The nonlinear variational problems for a function u with basis function v take the form

$$F(u, v) = 0, \tag{6.14}$$

where F is nonlinear, while the linear variational problems take the form

$$a(u, v) = L(v), \tag{6.15}$$

where a is bilinear and L linear. In Firedrake, these variational problems are defined with the $NonlinearVariationalProblem$ and $LinearVariationalProblem$ functions. The solvers corresponding to the symplectic-Euler and Störmer-Verlet weak formulations are then defined as follows:

```
#------------------------------------------------------------------------#
#                              Symplectic-Euler                          #
#________________________________________________________________________#
if scheme=="SE":
    #_______________ Variational solver for h (and hat_psi^*) _______________#
    h_problem = NonlinearVariationalProblem(WF_h_psi, w_n1)
    h_solver = NonlinearVariationalSolver(h_problem, solver_parameters=param_h)

    #_____________________ Variational solver for psi_1 _____________________#
    psi_problem = LinearVariationalProblem(A_psi_s, L_psi_s, psi_1_n1)
    psi_solver = LinearVariationalSolver(psi_problem, solver_parameters=param_psi)

    #____________________ Variational solver for hat_psi ____________________#
    hat_psi_problem = LinearVariationalProblem(A_hat, L_hat, hat_psi_n0)
    hat_psi_solver = LinearVariationalSolver(hat_psi_problem, solver_parameters=
    param_hat_psi)

#------------------------------------------------------------------------------#
#                               Stormer-Verlet                                 #
#______________________________________________________________________________#
if scheme=="SV":
    #_________ Variational solver for psi_1^{n+1/2} (and hat_psi^*) _________#
    psi_half_problem = NonlinearVariationalProblem(WF_psi_star, w_half)
    psi_half_solver = NonlinearVariationalSolver(psi_half_problem, solver_parameters=
    param_h)

    #________________ Variational solver for h^{n+1} psi_i^** _______________#
    h_problem = NonlinearVariationalProblem(WF_h_psi, w_n1)
    h_solver = NonlinearVariationalSolver(h_problem, solver_parameters=param_h)

    #__________________ Variational solver for psi_1^{n+1} __________________#
    psi_n1_problem = LinearVariationalProblem(a_psi_1, L_psi_1, psi_1_n1)
    psi_n1_solver = LinearVariationalSolver(psi_n1_problem, solver_parameters=
    param_psi)

    #____________________ Variational solver for hat_psi ____________________#
    hat_psi_problem = LinearVariationalProblem(A_hat, L_hat, hat_psi_n0)
    hat_psi_solver = LinearVariationalSolver(hat_psi_problem, solver_parameters=
    param_hat_psi)
```

Before solving the equations, files and functions are created to save the solutions. The saving settings are made in the file *savings.py* detailed next.

Saving functions

All the saved files described in section 6.3.2 are defined by calling the function *saving_files()* as follows:

```
"""
    ************************************************************
    *                       Saving Files                       *
    ************************************************************ """
def saving_files(save_path):
    save_waves = File(os.path.join(save_path, "waves.pvd"))
    save_WM = File(os.path.join(save_path, "Wavemaker.pvd"))
    WM_file = open(os.path.join(save_path, 'wm_motion.txt'), 'w')
    Energy_file = open(os.path.join(save_path, 'energy.txt'), 'w')
    README_file = open(os.path.join(save_path, 'README.txt'), 'w')
    return save_phi, save_WM, WM_file, Energy_file, README_file
```

The energy is saved through the call of the function *save_energy()* in which the energy is computed, and saves it in the text file *energy.txt*. As explained in Chapter 3, the unknowns h, ψ_1 and $\psi_{i'}$ are defined on the horizontal plane. In order to save the velocity potential $\phi(x, y, z, t)$, which is also a function of depth, the numerical domain on which the variables are defined must be extruded in depth. If the initial domain is 2D, then the solutions are only x–dependent, meaning that the mesh is 1D. As shown in Fig. 6.2, the functions are thus first expanded to a 2D horizontal mesh. Then, the 2D horizontal mesh is extruded in depth with height H_0 to obtain a 3D numerical domain with constant depth H_0, which is the transformed numerical domain defined in Chapter 3.

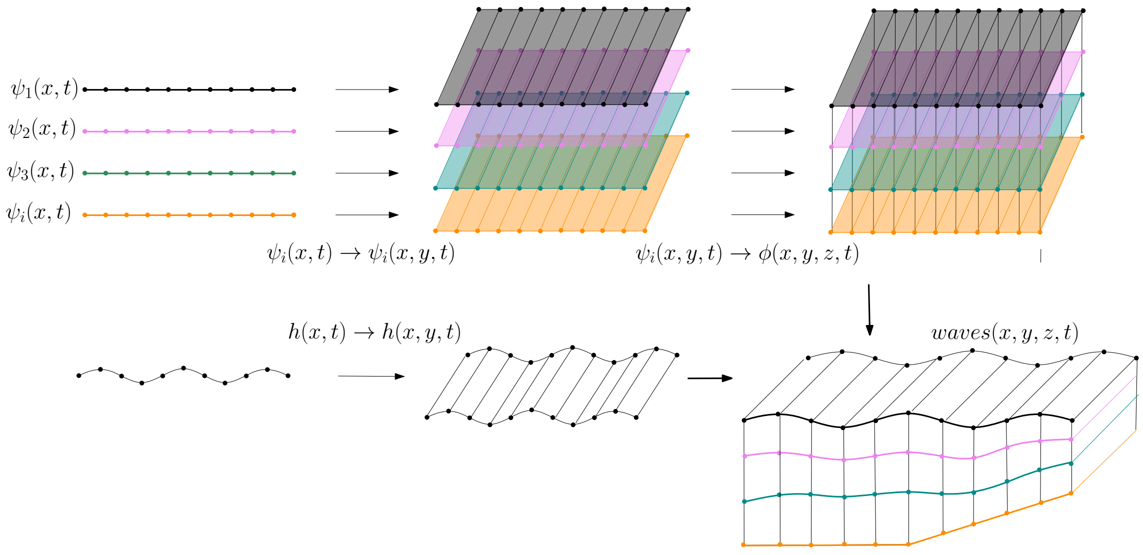


Figure 6.2: Mapping the x–dependent solutions to the 3D free-surface domain.

```
"""
    ****************************************************************************
    *                    Save waves in the 3D free-surface domain              *
    ****************************************************************************"""
#------------------------------------------------------------------------------------#
#                                    Saving mesh                                     #
#------------------------------------------------------------------------------------#
if dim=='2D': # Extend the 1D horizontal mesh (x) to 2D horizontal mesh (x,y)
```

```
    mesh_2D = RectangleMesh(Nx,1,Lx,Ly,quadrilateral=True)        # 2D surface mesh
    V_2D = FunctionSpace(mesh_2D,"CG",1)                         # 2D surface funct. space
    Vec_2D = VectorFunctionSpace(mesh_2D,"CG",1, dim=n_z)  # 2D vector funct. space
    h_2D = Function(V_2D)                                                     # h(x,y)
    psi_s_2D = Function(V_2D)                                            # psi_1 (x,y)
    psi_i_2D = Function(Vec_2D)                                          # psi_i (x,y)
    beach_s_2D = Function(V_2D).interpolate(beach_expr)                       # b(x,y)
    # Extend the surface mesh in depth to obtain {0<x<Lx; 0<y<Ly; 0<z<H0}
    mesh_3D = ExtrudedMesh(mesh_2D,                        # horizontal mesh to extrude;
                           n_z,                # number of elements in the vertical;
                           layer_height=H0/(n_z),         # length of each element;
                           extrusion_type='uniform')      # type of extruded coord.;

else:# If the solutions are already (x,y)-dependent, we extend the domain in depth:
    mesh_3D = ExtrudedMesh(hor_mesh,                       # horizontal mesh to extrude;
                           n_z,                # number of elements in the vertical;
                           layer_height=H0/(n_z),         # length of each element;
                           extrusion_type='uniform')      # type of extruded coord.;
```

The solution $waves$ which takes the value of $\phi(x, y, z, t)$ in the 3D domain is defined as:

```
"""
    *****************************
    *      Function to save     *
    ***************************** """
#_________ Function Space _________#
V_3D = FunctionSpace(mesh_3D, "CG",1)
#___________ Functions ____________#
waves = Function(V_3D,name="phi")
WM_3D = Function(V_3D,name = "WM")
```

In order to both map each ψ_i to the corresponding vertical layer and transform the 3D mesh back to the free-surface domain (for display purpose), the appropriate indices are selected as follows:

```
"""
    *************************************************************************
    *                          Mapping and transforms                       *
    *************************************************************************"""
if dim=="2D":
    # Indices to map h(x) and phi(x) to h(x,y) and phi(x,y) :
    Indx = []
    for j in range(len(hor_mesh.coordinates.dat.data[:])):
        Indx.append([y for y in range(len(mesh_2D.coordinates.dat.data[:,0]))\
         if mesh_2D.coordinates.dat.data[y,0]==hor_mesh.coordinates.dat.data[j]])

# Index used to differentiate each vertical layer
Indz = []
for i in range(0,n_z+1):
    Indz.append([zz for zz in range(len(mesh_3D.coordinates.dat.data[:,2])) \
     if mesh_3D.coordinates.dat.data[zz,2] == mesh_3D.coordinates.dat.data[i,2]])

# Index of the 3D funct. for which x<Lw. This is used to transform the 3D domain
# in x, to get back to the moving domain:
Test_x_Lw=Function(V_3D)
Test_x_Lw.interpolate(Expression('0.5*(1.0+copysign(1.0,Lw-x[0]))',Lw=Lw))
Indw = [item for item in range(len(Test_x_Lw.dat.data[:])) \
        if Test_x_Lw.dat.data[item] != 0.0]
```

Using $Indx$, $Indz$ and $Indw$, the solution $waves$ is saved at each time t in 3D. First, if the depth and velocity potential are defined on the x–plane only, the function *x_to_xy()* copies the 1D functions $h(x)$ and $\psi_i(x)$ to the surface plane (x, y) as follows:

```
#------------------------------------------------------------#
#                  Surface solutions (x,y)                   #
#------------------------------------------------------------#
def x_to_xy(h_n0, psi_1_n0, hat_psi_n0, h_2D, psi_s_2D, psi_i_2D, Indx):
    for i in range(len(h_n0.dat.data[:])):
        h_2D.dat.data[Indx[i]] = h_n0.dat.data[i]
        psi_s_2D.dat.data[Indx[i]]=psi_1_n0.dat.data[i]
        psi_i_2D.dat.data[Indx[i],:] = hat_psi_n0.dat.data[i,:]
```

Then, *Indz* is used to assign the velocity-potential values to each interior layer i, through the function *phi_projection()*:

```
#--------------------------------------------------------------------------#
#                          3D solution (x,y,z)                             #
#--------------------------------------------------------------------------#
def phi_projection(i, n_z, waves, Indz, psi_s, psi_i):
    if i==n_z:                                                    # if i=1,
        waves.dat.data[Indz[i]] = psi_s.dat.data[:]        # phi(z_i)=psi_1
    else:                                                         # if i>1,
        waves.dat.data[Indz[i]] = psi_i.dat.data[:,n_z-1-i]# phi(z_i)=psi_i
```

The 3D mesh is then transformed in z and x so that it moves with the free surface and wavemaker:

```
#------------------------------------------------------------------------------#
#                              Transform the domain                            #
#------------------------------------------------------------------------------#

#_________________________________ z-transform _________________________________#
def z_transform(mesh_3D, n_z, h_2D, beach_2D, H0, Indz):
    for i in range(0, n_z+1):                                      # for each layer
        mesh_3D.coordinates.dat.data[Indz[i],2]*=h_2D.dat.data[:]/H0   # z -> z*h/H0
        mesh_3D.coordinates.dat.data[Indz[i],2]+=beach_2D.dat.data[:] # z -> z+b(x)

#_________________________________ x-transform _________________________________#
def x_transform(mesh_3D, Lw, WM_3D, Indw):
    for i in range(0,len(Indw)):                              # x -> R + x*(Lw-R)/Lw
        mesh_3D.coordinates.dat.data[Indw[i],0]*=(Lw-WM_3D.dat.data[Indw[i]])/Lw
        mesh_3D.coordinates.dat.data[Indw[i],0]+=WM_3D.dat.data[Indw[i]]
```

Time loop

The above functions are called in the main time loop to save the energy and *waves* in the free-surface domain that moves with $h(x, y, t)$ and $R(y, t)$. The numerical domain is then transformed back to the fixed coordinates in order to update the solutions by calling the solvers in the fixed domain:

```
t_save = t
while t<Tend-dt:
    """ **********************************************************************
        *                            SAVE FUNCTIONS                          *
        ********************************************************************** """
    if t_save <= t:
        print('Progress: ', 100*t/Tend, ' %')
        #------------------------------------------------------------------------#
        #                                 ENERGY                                 #
        #------------------------------------------------------------------------#
        save_energy(h_n0, psi_1_n0, hat_psi_n0, WM, dWM_dy, dWM_dt, H, x_coord, Lw,
                    H0, g, A11, AN1, A1N, ANN, M11, M1N, MN1, MNN, D11, D1N, DN1,
                    DNN, S11, S1N, SN1, SNN, I1, IN, Energy_file, t, dim)


        #------------------------------------------------------------------------#
        #                           SAVE 3D FUNCTIONS                            #
        #------------------------------------------------------------------------#
        #___________________________ Project solutions ___________________________#
        if dim == ’2D’:
            # To the surface plane (x,y) :
            x_to_xy(h_n0, psi_1_n0, hat_psi_n0, h_2D, psi_s_2D, psi_i_2D, Indx)
            # In depth (x,y,z):
            for i in range(0,n_z+1):                  # for each layer phi(z)=psi_i
                phi_projection(i, n_z, waves, Indz, psi_s_2D, psi_i_2D)
                WM_3D.dat.data[Indz[i]] = WM.dat.data[0]               # WM(z) = WM
        elif dim == ’3D’:
            # In depth (x,y,z):
            for i in range(0,n_z+1):
                phi_projection(i, n_z, waves, Indz, psi_1_n0, hat_psi_n0)
                WM_3D.dat.data[Indz[i]] = WM.dat.data[:]

        #_______________________ Save the fixed coordinates ______________________#
        init_coord = mesh_3D.coordinates.vector().get_local()

        #______________________________ z-transform ______________________________#
        if dim == ’2D’:
            z_transform(mesh_3D, n_z, h_2D, beach_s_2D, H0, Indz)
        elif dim == ’3D’:
            z_transform(mesh_3D, n_z, h_n0, beach, H0, Indz)

        #______________________________ x-transform ______________________________#
        x_transform(mesh_3D, Lw, WM_3D, Indw)

        #______________________________ Save waves _______________________________#
        save_waves.write(waves)

        #_______________________ Back to the initial mesh ________________________#
        mesh_3D.coordinates.vector().set_local(init_coord)

        #____________________________ Save wavemaker _____________________________#
        save_WM.write(WM_3D)

        #__________________________ Update saving time ___________________________#
        t_save+=dt_save
```

Subsequently, the time is updated. Depending on the initial settings, the wavemaker is also updated if $t < t\_stop$. However, as soon as $t > t\_stop$ the wavemaker is switched off and does not need to be updated anymore. The parameter *update_wm* is initialised to ’Yes’ and set to ’No’ once the wavemaker has been switched off, so that after the wavemaker stops these transforms are not needed every time step.

```
""" ****************************************************************
    *                   Update time and wavemaker                  *
    **************************************************************** """
    #_____________________ Update time: t^n -> t^{n+1} __________________#
    t_half = t+0.5*dt
    t += dt
    #________________________ Update wavemaker ________________________#

    if t<=t_stop:                           # The wavemaker keeps moving
        if scheme=="SV":
            WM_expr.t = t_half
            dWM_dt_expr.t = t_half
            dWM_dy_expr.t = t_half
            WM_half.interpolate(WM_expr)                # update R(x,y;t)
            dWM_half_dt.interpolate(dWM_dt_expr)           # update dR/dt
            dWM_half_dy.interpolate(dWM_dy_expr)           # update dR/dy
        WM_expr.t = t
        dWM_dt_expr.t = t
        dWM_dy_expr.t = t
        WM_n1.interpolate(WM_expr)                      # update R(x,y;t)
        dWM_n1_dt.interpolate(dWM_dt_expr)                # update dR/dt
        dWM_n1_dy.interpolate(dWM_dy_expr)                # update dR/dy
        t_aux = t

    elif t>t_stop and update_wm=='Yes': # We stop the wavemaker motion;
        update_wm='No'
        if scheme=="SV":
            if t_half<=t_stop:
                t_aux = t_half
            WM_expr.t = t_aux
            dWM_dt_expr.t = t_aux
            dWM_dy_expr.t = t_aux
            WM_n1.interpolate(WM_expr)
            dWM_n1_dt.assign(0.0)
            dWM_n1_dy.interpolate(dWM_dy_expr)
            if scheme=="SV":
                WM_half.interpolate(WM_expr)
                dWM_half_dt.assign(0.0)
                dWM_half_dy.interpolate(dWM_dy_expr)
```

Note that for the Störmer-Verlet scheme, the wavemaker motion and derivatives also needs to be computed at time $t^{n+1/2}$. Once time is updated, the solvers are called through:

```
""" *****************************************************
    *               Solve the weak formulations           *
    ******************************************************* """
    #___________________ Call the solvers ___________________#
    if scheme=="SE":                                # 1st-order SE scheme
        h_solver.solve()                # get h^{n+1} and hat_psi^*
        psi_solver.solve()                          # get psi^{n+1}
    elif scheme=="SV":                              # 2nd-order SV scheme
        psi_half_solver.solve()# get psi^{n+1/2} and hat_psi^*
        h_solver.solve()            # get h^{n+1} and hat_psi^{**}
        psi_n1_solver.solve()                       # get psi_1^{n+1}
```

Finally, the functions are updated for the next time step.

```
""" ******************************
    *    Update the functions    *
    ****************************** """
    #_____ Update the solutions _____#
    h_out, hat_psi_out = w_n1.split()
    h_n0.assign(h_out)
    psi_1_n0.assign(psi_1_n1)
    hat_psi_n0.assign(hat_psi_out)

    #_____ Update the wavemaker _____#
    WM.assign(WM_n1)
    dWM_dt.assign(dWM_n1_dt)
    dWM_dy.assign(dWM_n1_dy)
```

All the parameters used to obtain the simulations are saved on a README file at the end of the simulations. This file takes the form:

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                    Summary
------------------------------------------------
-------- Dimensions of the domain --------
Length Lx: 10.0  m
Length Ly: 1.0   m
Depth H0: 1.0   m
Beach start: 3.0   m
Beach slope: 0.1

-------------- Mesh resolution --------------
In x: 0.05  m (200.0 elements)
In y: 0.05  m (20.0  elements)
In z: 0.125 m (8     elements)

----------------- Wavemaker -----------------
Amplitude: 0.02  m
Period: 1.13  s
Frequency: 5.55752212389 /s
Stops after 5.0   periods ( = 5.65  s)
Lw: 1.0   m

------------------- Solver -------------------
1st order Symplectic-Euler scheme

---------------- Final time ----------------
Tend: 11.3
dt: 0.001
Computational time: 0 j 0 h 10 mn 36 s (=636 s)
```

In next section, we explain how to couple the deep-water tank to a shallow-water beach.

6.4 Numerical wave tank: coupling to shallow-water beach

6.4.1 Introduction

In Chapter 4, a coupling between the deep-water potential-flow equations and the shallow-water equations was derived to absorb the waves on a beach. As discussed, the potential-flow equations have been solved with the finite-element method and their implementation has been described in section 6.3. In this section, we describe the finite-volume implementation of the shallow-water solvers and explain how to couple these solvers to the potential-flow solvers.

6.4.2 Code source

In addition to the files used for the 3D tank (*cf.* section 6.3), a file *NLSW_beach.py* is used to define all the functions related to the nonlinear shallow-water beach implementation. Moreover, the nonlinear potential-flow weak formulations as given in section 6.3 need to be extended with coupling terms. A file *NLDW_WM.py* containing the nonlinear deep-water coupled weak formulations is thus used instead of the *solvers.py* file when considering the coupled tank.
The codes return several functions that can be opened in Paraview.

- The files $dw\_waves.pvd$ and $sw\_waves.pvd$ contain the deep-water velocity potential $\phi(x, z, t)$ and the shallow-water velocity $u(x, t)$ respectively, saved in the deep- and shallow-water domains respectively. In order to observe their evolution, open the two files in Paraview, and apply a "Gradient on unstructured dataset" filter to the $dw\_waves.pvd$. This filter enables one to observe the spatial derivatives of ϕ and therefore to get the deep-water velocity. Subsequently, choose to fill the domain with "sw_u" for the function $sw\_waves.pvd$ and "Gradients X" for the function $dw\_waves.pvd$. Set the same colorbar scale for the two variables, and play to observe the solutions. In the movies, you should observe continuous free-surface and velocity fields throughout the coupled domain. You can also open the file *wavemaker.pvd* to see the wavemaker motion.
- To observe only the free-surface evolution, open the files $dw\_h.pvd$, $sw\_h.pvd$, $dw\_beach.pvd$ and $sw\_beach.pvd$. Apply a "Wrap by scalar" filter on all variables, and

you will observe the free-surface evolution above the sea-bed topography.

- The deep-water and shallow-water energies are saved in the files $dw\_energy.txt$ and $sw\_energy.txt$ respectively. Open these files in Matlab or Python for example to plot the energy of each domain as a function of time.
- Finally, the file ”README.txt” summarizes the test parameters.

In the next section, instructions are given on how to run the coupled model.

6.4.3 Use of the code

First of all, select the input data, either using the choice “*created*” or “*measurements*” in the function *input()*. In this tutorial, we only consider the case of created data. The comparison with measurements is explained in section 6.5.

```
"""
    ********************************************
    *                 Test case                *
    ********************************************"""
def input():
    #________________ Kind of data ________________#
    #input_data = "measurements"   # from experiments
    input_data = "created"      # set the wavemaker
    #_________________ Test case __________________#
    if input_data == "created":
        test_case = "test"     # choose a folder name
    elif input_data == "measurements":
        test_case = "111"         # choose from table
    return input_data, test_case
```

In the case of created data, the variable *test_case()* is used to specify the folder in which the numerical data will be saved.

In the coupled model, the domain length is increased so that the beach goes above the depth at rest; the beach length must be set to that waves do not reach the end of the domain. In this example, we let the beach go 20% above the rest waterline, but this length can be changed depending on the wave amplitudes. The characteristics of the coupled model only depend on the deep-water settings, defined through the function *domain()*, as explained in section 6.3. In order to be coupled to the shallow-water model as accurately as possible, the depth at the end of the deep-water domain

$H_c = H(x_c)$ must satisfy

$$\lambda/20 \ll H_c \ll \lambda/2, \tag{6.16}$$

so make sure to define H_c so that the above criteria is satisfied. The deep-water domain characteristics are imported into the main program as

```
#------------------------- Deep-water domain -------------------------#
H0, xb, slope, H_expr, Hc, xc, Lw, res_dw, n_z = domain(input_data)
```

so that the coupling coordinate x_c is defined at the end of the deep-water domain, where the depth at rest is $H_c = H(x_c)$. The shallow-water domain is defined from x_c to L, where L is the length necessary for the end of the domain to lie a fraction f_L, say 20%, above the maximal depth at rest $H_0 = H_r(x \leq x_B)$ (*cf.* Chapter 4). The length L_{sw} of the shallow-water domain is thus obtained directly from the deep-water characteristics, as $L_{sw} = L - x_c$. Similarly, the resolution in the shallow-water domain is defined as

$$\Delta x_{SW} = \Delta x_{DW}^2, \tag{6.17}$$

to account for the second-order accuracy in the deep-water domain and first-order accuracy in the shallow water domain, as explained in Chapter 4.
The shallow-water and coupled domains lengths are therefore automatically computed in the main file, through:

```
#---- Shallow-water domain ----#
Lsw = (Hc+f_L*H0)/slope
res_sw = res_dw*res_dw

#---- Total coupled domain ----#
L_total = xc + Lsw
```

Similarly to the 3D wave tank case, specify the wavemaker and time parameters in the file $Settings.py$. Now we can run the main file to create a simulation.

6.4.4 Code description

In this section, we detail the extensions of the deep-water code described in section 6.3 that are necessary to obtain the coupled simulations. In particular, we explain how the shallow-water numerical domain is defined, how the finite-volume solvers are implemented, and how the deep-water solvers are coupled to the shallow-water ones.

Definition of the shallow-water solutions

First, the shallow-water mesh is defined, similarly to the deep-water mesh.

```
"""
    *********************************************
    *           Definition of the mesh          *
    ********************************************* """
#___________ Horizontal discretization ____________#
Nv_sw = round(L_sw/res_sw)  # Number of volumes in x
#_______________ Shallow-water mesh _______________#
sw_mesh = IntervalMesh(Nv_sw,L_sw)         # SW mesh
```

Note that the shallow-water x–coordinate is shifted as $\check{x} = x - x_c$ in order to start at $\check{x} = 0$. As explained in Chapter 4, this enables to define the shallow-water topography as

$$\check{H}(x) = H_c - \check{b}(\check{x}), \quad \text{with} \quad \check{b}(\check{x}) = s_B\check{x}. \tag{6.18}$$

The shallow-water solutions h and hu are discontinuous, so the function space is defined based on discontinuous-Galerkin expansions, through the option "$DG0$":

```
#__________ Function Space __________#
sw_V = FunctionSpace(sw_mesh,"DG", 0)

#_____________ Functions _____________#
h_fv = Function(sw_V, name="h_fv")
hu_fv = Function(sw_V, name = "hu_fv")
sw_beach = Function(sw_V)
```

As we define the finite-volume solvers by hand (*ie.* not using Firedrake), no trial or basis function needs to be defined. The functions h and hu defined on the shallow-water mesh will only be used to save the shallow-water solutions. In addition, the function hu_fe is introduced to apply the deep-water flux boundary conditions. It is defined in the deep-water function space *dw_V* as follows:

```
# BCs for deep-water:
hu_fe = Function(dw_V)
```

The solutions are initialised at the rest state. By definition, the discrete topography $\check{b}_k$ is the averaged topography across volume X_k:

$$\check{b}_k = \frac{1}{\Delta x_{SW}} \int_{x_{k-1/2}}^{x_{k+1/2}} \check{b}(\check{x}) \mathrm{d}\check{x}, \quad \text{for} k = 1, \dots Nv_{sw}. \tag{6.19}$$

Therefore, it is nonzero at the coupling point $x = x_c$. In order to ensure continuity of the rest depth accross the interface, the rest depth at the coupling $x = x_c^+$ is then defined as

$$H(x_c^+) = H_0 - H(x_c^-) - \check{b}_0, \tag{6.20}$$

where $\check{b}_0$ is the discrete topography in the first volume X_0. The initial depth h_fv is then initialised as the rest depth, that is

$$h_k = H(x_c^+) - \check{b}_k. \tag{6.21}$$

Hence, in the code, the functions are initialised through:

```
#--------------------------- Shallow-water beach ---------------------------#
beach_sw_expr = Expression("slope*x[0]",slope=slope)    #\check{b}(\check{x})
H0_sw = H0-sw_beach.dat.data[0]-dw_beach.dat.data[-1]             # H(x_c^+)
sw_beach.interpolate(beach_sw_expr)
#--------------------------- Shallow-water depth ---------------------------#
h_fv.assign(H0_sw-sw_beach)                   # h = H(x_c^+)-\check{b}(\check{x})
h_fv.dat.data[np.where(h_fv.vector().get_local()<0)]=0 # avoid negative depth
#------------------------- Shallow-water velocity --------------------------#
hu_fv.assign(0.0)                                                 # u(t=0)=0
```

Definition of the shallow-water solver

The shallow-water solver is defined in separation of Firedrake in the file *NLSW_beach.py* and is called at each time step. First, the Firedrake functions h_fv and hu_fv are copied into the two-column array U that will be used as the solution at time t^n. Similarly, the beach topography is saved in the array bk:

```
#--------- Shallow-water solutions ---------#
U = 0*np.eye(2,Nvol+1)
U[0,1:Nvol+1] = h_fv.dat.data[0:Nvol]   # h^n
U[1,1:Nvol+1] = hu_fv.dat.data[0:Nvol] # hu^n
#----------- Shallow-water beach -----------#
```

```
bk = 0*np.eye(1,Nvol+1)
bk[0,1:Nvol+1] = sw_beach.dat.data[:]   # b(x)
```

Then, the boundary values for h and hu, computed from the values of the deep-water solutions (*cf.* Chapter 4), are assigned to the ghost cells of the shallow-water domain:

```
#--------------- Depth ---------------#
U[0,0]=h_bc             # Left: coupling
U[0,-1]=U[0,-2]             # Right: h=0

#------------- Velocity --------------#
U[1,0]=hu_bc            # Left: coupling
U[1,-1]=-U[1,-2]       # Right: du/dx =0

#--------------- Beach ---------------#
bk[0,0] = bk[0,1]-(bk[0,2]-bk[0,1])
bk[0,-1] = bk[0,-2]-(bk[0,-3]-bk[0,-2])
```

The indices -1 and -2 designate the last and penultimate volumes respectively. The updated solution is initialised in the vector U_next as follows:

```
U_next=np.copy(U)
```

The numerical flux at the interface denoted by $x_{k+1/2}$ requires the values in the adjacent cells k and $k + 1$. We assign the values of U_k, U_{k-1} and U_{k+1} as follows:

```
k = range(1,Nvol-1)
k_plus = range(2,Nvol)
k_minus = range(0,Nvol-2)
Uk = np.copy(U[:,k])              # U_k
Uk_plus = np.copy(U[:,k_plus])    # U_{k+1}
Uk_minus = np.copy(U[:,k_minus]) # U_{k-1}
```

The left and right depths at each volume interface are then computed using the method introduced by Audusse [5] to ensure non-negative depth (*cf.* Eq. 4.32 in Chapter 4 for more details):

```
#--------------------------- Non-negative depth --------------------------#
bk_half_r = np.maximum(bk[0,k],bk[0,k_plus])                     # b_{k+1/2}
bk_half_l = np.maximum(bk[0,k],bk[0,k_minus])                    # b_{k-1/2}
h_plus_r = np.maximum(Uk_plus[0,:]+bk[0,k_plus]-bk_half_r,0)   # h_{k+1/2^+}
h_plus_l = np.maximum(Uk[0,:]+bk[0,k]-bk_half_r,0)             # h_{k+1/2^-}
h_minus_r = np.maximum(Uk[0,:]+bk[0,k]-bk_half_l,0)            # h_{k-1/2^+}
h_minus_l = np.maximum(Uk_minus[0,:]+bk[0,k_minus]-bk_half_l,0)# h_{k-1/2^-}
```

Finally, the interface value of hu is computed and saved to the vectors U_plus_l ($U_{k+1/2^-}$), U_plus_r ($U_{k+1/2^+}$), U_minus_l ($U_{k-1/2^-}$) and U_minus_r ($U_{k-1/2^+}$) by calling the function *U_half* that returns

$$U_{k-1/2^-} = \begin{pmatrix} h_{k-1/2^-} \\ h_{k-1/2^-}u_{k-1} \end{pmatrix}, \quad U_{k-1/2^+} = \begin{pmatrix} h_{k-1/2^+} \\ h_{k-1/2^+}u_k \end{pmatrix}, \tag{6.22}$$

$$U_{k+1/2^-} = \begin{pmatrix} h_{k+1/2^-} \\ h_{k+1/2^-}u_k \end{pmatrix}, \quad U_{k+1/2^+} = \begin{pmatrix} h_{k+1/2^+} \\ h_{k+1/2^+}u_{k+1} \end{pmatrix}. \tag{6.23}$$

```
#----- Left and right values of U at the interface k+/-1/2 -----#
[u_plus_l,U_plus_l] = U_half(h_plus_l, Uk)             # U(k+1/2)-
[u_plus_r,U_plus_r] = U_half(h_plus_r, Uk_plus)        # U(k+1/2)+
[u_minus_l,U_minus_l] = U_half(h_minus_l, Uk_minus)    # U(k-1/2)-
[u_minus_r,U_minus_r] = U_half(h_minus_r, Uk)          # U(k-1/2)+
```

The corresponding left and right fluxes of the cell boundary $x_{k+1/2}$, $F_L(U_{k+1/2})$ and $F_R(U_{k+1/2})$ respectively, are then computed from the definition of the flux

$$F(U) = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{pmatrix} \tag{6.24}$$

with the function *flux_half()* so that

$$F_L(U_{k+1/2}) = \begin{pmatrix} h_{k+1/2^-}u_k \\ h_{k+1/2^-}u_k^2 + \frac{1}{2}gh_{k+1/2^-}^2 \end{pmatrix}, \tag{6.25}$$

$$\text{and } F_R(U_{k+1/2}) = \begin{pmatrix} h_{k+1/2^+}u_{k+1} \\ h_{k+1/2^+}u_{k+1}^2 + \frac{1}{2}gh_{k+1/2^+}^2 \end{pmatrix}. \tag{6.26}$$

They are assigned to the corresponding variables Fl_plus ($F_L(U_{k+1/2})$) and Fr_plus ($F_R(U_{k+1/2})$) as follows:

```
# Left and right values of F at the interface k+1/2 #
Fl_plus = flux_half(U_plus_l,g_tilde)   # Fl(U_{k+1/2})
Fr_plus = flux_half(U_plus_r,g_tilde)   # Fr(U_{k+1/2})
```

As all values are assigned at once, with vectorial assignment, we also define the left and right fluxes at the boundary $x_{k-1/2}$ using

```
# Left and right values of F at the interface k-1/2 #
Fl_minus =flux_half(U_minus_l,g_tilde)  # Fl(U_{k-1/2})
Fr_minus = flux_half(U_minus_r,g_tilde) # Fr(U_{k-1/2})
```

In addition, the left and right speeds at each cell boundary must be computed to estimate the HLL flux (C.46). The wave speeds are defined as

$$S_L(U_{k\pm1/2}) = \min\left(u_{k\pm1/2-} - \sqrt{gh_{k\pm1/2-}}, u_{k\pm1/2+} - \sqrt{gh_{k\pm1/2+}}\right), \tag{6.27a}$$

$$S_R(U_{k\pm1/2}) = \max\left(u_{k\pm1/2-} + \sqrt{gh_{k\pm1/2-}}, u_{k\pm1/2+} + \sqrt{gh_{k\pm1/2+}}\right). \tag{6.27b}$$

Therefore, we define a function *left_wave_speed()* and a function *right_wave_speed()* as follows:

```
def left_wave_speed(ul, ur, hl, hr, g):
    Sl = np.minimum(ul- np.sqrt(g*hl), ur- np.sqrt(g*hr))
    return Sl

def right_wave_speed(ul, ur, hl, hr, g):
    Sr = np.maximum(ul+ np.sqrt(g*hl), ur+ np.sqrt(g*hr))
    return Sr
```

where u_L, u_R, h_L, h_R are the left and right limits of u and h at the considered boundary. Namely, the left and right wave speeds of each boundary are assigned through

```
#----------- Left and right speeds at each interface of cell k ----------#
# Sl(U_{k-1/2}): Ul/r=U(k-1/2)-/+
Sl_minus=left_wave_speed(u_minus_l,u_minus_r,h_minus_l,h_minus_r,g_tilde)
# Sr(U_{k-1/2}): Ul/r=U(k-1/2)-/+
Sr_minus=right_wave_speed(u_minus_l,u_minus_r,h_minus_l,h_minus_r,g_tilde)
# Sl(U_{k+1/2}): Ul/r=U(k+1/2)-/+
Sl_plus=left_wave_speed(u_plus_l,u_plus_r,h_plus_l,h_plus_r,g_tilde)
# Sr(U_{k+1/2}): Ul/r=U(k+1/2)-/+
Sr_plus=right_wave_speed(u_plus_l,u_plus_r,h_plus_l,h_plus_r,g_tilde)
```

The HLL fluxes at each flux boundary $F_{k-1/2}$ and $F_{k+1/2}$ are finally assigned to their respective variables F_minus and F_plus through

```
#---------------------------------- HLL fluxes ----------------------------------#
# F(k-1/2)  = F(Fl,Fr) ; Fl  = F(U(k-1/2)-) ; Fr  = F(U(k-1/2)+)
F_minus = HLL_flux(Fl_minus, Fr_minus,  Sl_minus, Sr_minus, U_minus_l, U_minus_r)
# F(k+1/2)  = F(Fl,Fr) ; Fl  = F(U(k+1/2)-) ; Fr  = F(U(k+1/2)+)
F_plus  = HLL_flux(Fl_plus, Fr_plus, Sl_plus, Sr_plus, U_plus_l, U_plus_r)
```

where the function *HLL_flux()* computes the HLL flux (C.46) depending on the left and right wave speeds and left and right fluxes.

Note that calculation of the flux (and therefore wave speed) at the boundary $x_{k-1/2}$ can be avoided by looping over the cell interfaces instead, in which case $F_{k-1/2}$ takes the value of $F_{k+1/2}$ at the previous iteration.

The depth h_k may then be updated by solving the conservation of mass shallow-water equation, that is,

```
#-------------------------------- Update h --------------------------------#
U_next[0,k] = Uk[0,:] - dt*(F_plus[0,:] - F_minus[0,:])/d_x
```

Knowing h at time t^{n+1}, the above steps are repeated to update hu, but this time using h_k^{n+1} instead of h_k^n, in order to obtain the symplectic-Euler scheme. After a consequent update of the numerical fluxes and computation of the topography term through

```
#-------------------------- Topography --------------------------#
Sk[1,:]=0.5*g_tilde*h_next_plus_l**2-0.5*g_tilde*h_next_minus_r**2
```

the solution hu_k is updated through:

```
#-------------------------------- Update hu -------------------------------#
U_next[1,k] = Uk[1,:] - dt*(F_plus[1,:] - F_minus[1,:])/d_x + dt*Sk[1,:]/d_x
```

The flux at $x = x_c$ is saved in the variable *hu_fe* in order to be used as boundary condition for the deep-water equations. In addition, the updated vector U_next containing the solutions h and hu at time t^{n+1} is assigned to the Firedrake functions h_fv and hu_fv so that they can be saved in the main file. Finally, the shallow-water energy is computed and returned to the main file:

```
# BC for DW: flux at the left boundary
hu_fe.vector().set_local(F_minus[0,0])
#-------- Update the solutions -------#
h_fv.vector().set_local(U[0,1:Nvol+1])
hu_fv.vector().set_local(U[1,1:Nvol+1])
#---- Compute the energy ----#
hu_square = 0.0*np.eye(1,Nvol)
h_square = 0.0*np.eye(1,Nvol)
kk = range(1,Nvol)
kk_minus = range(0,Nvol-1)
huu = 0.0*np.eye(1,len(U[0,:]))
[Ind] = np.where(U[0,:]>=1e-9)
```

```
huu[0,Ind] =U[1,Ind]*U[1,Ind]/U[0,Ind]
hu_square[0,kk_minus]=huu[0,kk]
h_square[0,kk_minus]=U[1,kk]*U[1,kk]
E_sw = d_x*(0.5*sum(hu_square[0,:]) + 0.5*g*sum(h_square[0,:]))
return h_fv, hu_fv, U, hu_fe, E_sw
```

In the next section, we detail the algorithm used to couple this shallow-water solver to the deep-water one.

Coupling the deep- and shallow-water solvers

First of all, the shallow-water flux hu_fe is applied at the boundary $x = x_c$ of the deep-water weak formulations. This is done weakly by adding terms involving hu_fe at the deep-water weak formulations boundaries, as explained in Chapter 4. The weak formulations are defined in the file *NLDW_WM.py* and imported through

```
"""
    ***************************************************************************
    *                       Define the deep-water solvers                     *
    *************************************************************************** """
#____________________ Variational problem for h (and hat_psi^*) _____________________#
DW_VP_h = dw.VP_h(Lw, WM, p, h_n1, h_n0, dt, H0, psi_s_n0, x_coord, dWM, hat_psi_aux,
    r, w_n1, hu_fe, A11, A1N, AN1, ANN, M11, M1N, MN1, MNN, D11, D1N, DN1, DNN, S11,
    S1N, SN1, SNN, I1, IN, I3, n_z)

#_________________________ Variational solver for psi_1 __________________________#
DW_VP_psi_s = dw.VP_psi_s(Lw, H0, g, dt, WM, WM_n1, dWM, v, x_coord, psi_s_n0, h_n1,
    psi_s_n1, psi_s, H, hat_psi_aux, A11, A1N, AN1, ANN, M11, M1N, MN1, MNN, D11, D1N,
     DN1, DNN, S11, S1N, SN1, SNN, I1, IN, I3, G1, GN, hu_fe)
```

The deep-water solvers are then defined as in section 6.3 through

```
"""
    ***************************************************************************
    *                       Define the deep-water solvers                     *
    *************************************************************************** """
param_h={"ksp_converged_reason":True,"pc_type": "fieldsplit",\
    "pc_fieldsplit_type": "schur","pc_fieldsplit_schur_fact_type":"upper"}
param_psi={"ksp_converged_reason":True}
param_hat_psi={"ksp_converged_reason":True,'ksp_type': 'preonly','pc_type':'lu'}

```

```
#____________________ Variational solver for h (and hat_psi^*) ____________________#
DW_solver_h = NonlinearVariationalSolver(DW_VP_h, solver_parameters=param_h)

#__________________________ Variational solver for psi_1 __________________________#
DW_solver_psi_s = LinearVariationalSolver(DW_VP_psi_s, solver_parameters=param_psi)
```

The algorithm as presented in Chapter 4 is then implemented as follows. After initialisation of the solutions, including the boundary flux hu_fe, the deep-water equations are solved:

```
""" ****************************************************
    *             Solve the weak formulations          *
    **************************************************** """
#______________ Call the deep-water solvers ______________#
DW_solver_h.solve()                         # h^{n+1}, psi_i^*
DW_solver_psi_s.solve()                          # psi_1^{n+1}
```

Then, the solutions h^{n+1}, $\psi_{i'}^*$ and ψ_1^n are used to compute the shallow-water boundary conditions, derived in Chapter 4:

$$\begin{aligned} h\_bc &= h^{n+1}_{x=x_c} \\ hu\_bc &= \frac{1}{H_0}\left[h^{n+1}\left(\partial_x\psi_1^n\tilde{I}_1 + \partial_x\psi_{i'}\tilde{I}_{i'}\right) - \partial_x h^{n+1}\left(\tilde{G}_1\psi_1^n + \tilde{G}_{i'}\psi_{i'}\right)\right]_{x=x_c} \end{aligned} \tag{6.28}$$

The above equations are assigned to the shallow-water boundaries through

```
#________ Update the boundary-condition solutions _______#
h_out, hat_psi_out = w_n1.split()

hu_bc = assemble((1/H0)*(h_out*psi_s_n0.dx(0)*I1 \
                         + h_out*dot(hat_psi_out.dx(0),IN)\
                         - G1*psi_s_n0*h_out.dx(0) \
                         - h_out.dx(0)*dot(GN,hat_psi_out))*ds(2))
h_bc = assemble((h_out)*ds(2))
```

The shallow-water equations can then be solved by calling the shallow-water solver defined in the previous paragraph, through

```
#______________ Call the shallow-water solver ______________#
h_fv, hu_fv, U, hu_fe, E_sw = sw.solve_FV(int(Nv_sw), res_sw, dt, sw_beach, g, h_bc,
    hu_bc, h_fv, hu_fv, hu_fe)
```

This way, the depth and velocity have been updated in the whole domain. The flux hu_fe returned by the shallow-water solver will be used as boundary flux for the next call of the deep-water solver.

In next section, we explain how the solutions are saved to be visualised in Paraview.

Saving the shallow-water solutions

In order to visualise the free surface and the waves velocity in shallow water, the shallow-water mesh is extruded in depth, just as the deep-water mesh described in section 6.3. The shallow-water free surface is saved on the $1D$-mesh through the function sw_h, in the file $sw\_h.pvd$. The shallow-water velocity is saved in the file $sw\_waves.pvd$, in which the shallow-water domain evolves with the free surface h_fv and is filled with the solution sw_u, defined in the vertical mesh.

```
#-------------------- Extruded vertical mesh --------------------#
sw_mesh_2D = ExtrudedMesh(sw_mesh, num_element, layer_height = H0,
 extrusion_type='uniform')
sw_V_2D = FunctionSpace(sw_mesh_2D, "DG", 0)
#---------------------- Vertical solutions ----------------------#
sw_u = Function(sw_V_2D, name = "sw_u")
sw_h = Function(sw_V, name="sw_h")
#------------------------ Saving files ------------------------#
sw_beach_file = File(os.path.join(save_path,"sw_beach.pvd"))
sw_waves_file = File(os.path.join(save_path,"sw_waves.pvd"))
sw_h_file = File(os.path.join(save_path,"sw_h.pvd"))
E_file_sw = open(os.path.join(save_path,"energy_sw.txt"), 'w')
```

To move with the free surface, the indices of the z–coordinates to transform are accessed through *sw_Indz*, just as those of the deep-water domain. In addition, the x–coordinate is shifted as $x \to x + x_c$ in order to start at the coupling point $x = x_c$, and the z–coordinate is shifted through $z \to z + H_0 - H_c + \check{b}(x_c)$, where $\check{b}(x_c)$ is the discontinuous value of the shallow-water beach, to account for the fact the the shallow-water domain starts at x_c with rest depth H_c. The solutions are thus saved as follows:

```
#------------------------------ Save h ------------------------------#
init_mesh = sw_mesh.coordinates.vector().get_local()
sw_mesh.coordinates.dat.data[:]+=xc
sw_h.assign(sw_beach.dat.data[0]+dw_beach.dat.data[-1] + sw_beach + h_fv)
sw_h_file.write(sw_h)
sw_mesh.coordinates.vector().set_local(init_mesh)
#--------------- Save u in the free-surface domain ---------------#
init_SW_coord = sw_mesh_3D.coordinates.vector().get_local()
```

```
#---- z transform
for i in range(len(h_fv.dat.data[:])):
    sw_waves.dat.data[i]=hu_fv.dat.data[len(h_fv.dat.data[:])-1-i]/h_fv.dat.data[len(
    h_fv.dat.data[:])-1-i]
    sw_mesh_3D.coordinates.dat.data[Indz_sw[i],2]*=h_fv.dat.data[i]
    sw_mesh_3D.coordinates.dat.data[Indz_sw[i],2]+=(sw_beach.dat.data[i]+dw_beach.dat.
    data[-1]+sw_beach.dat.data[0])
sw_mesh_3D.coordinates.dat.data[ii,2] = slope*(L_total-xb)
sw_waves_file.write(sw_waves)
#---- transform back
sw_mesh_3D.coordinates.vector().set_local(init_SW_coord)
```

The deep-water $waves$ solution is saved as explained in section 6.3. The deep-water depth is also saved in the file $dw\_h.pvd$ through

```
#------------------- Save DW h --------------------#
dw_h_file = File(os.path.join(save_path,"dw_h.pvd"))
dw_h_file.write(dw_h)
```

The solutions can be observed on Paraview, as explained in section 6.4.2.

6.5 Use of experimental data

6.5.1 Introduction

In this section we explain how to import measured wavemaker motion and velocity into the numerical simulations. In particular, we explain how to import the data collected in Chapter 5 and how they are treated before being used as input of the coupled tank. We also present the function used to compare the temporal and Fourier spectra of the numerical and experimental data. Finally, we detail the modifications of the code necessary to take into account the measured wavemaker.

6.5.2 Code source

The measurements may be loaded from the folder *Experimental data*. This folder contains

- a matlab function *make_input_file.m*, used to load and filter the measured data; the functions *toolbox_bandpass.m* and *toolbox_matfft.m*, provided by MARIN, are used to filter the measured data;

- a folder *Beach tests* containing all the test cases with the beach, referenced by the *test_case* number, as follows:

test case	run numbers	Period T_p [s]	Wave height H_s [m]
111	124 164	1.67	0.05
112	126 150		0.1
113	128 166		0.2
121	130 152	1.13	0.05
122	132 168		0.07
123	134 170		0.1
131	132 168	0.8	0.03
132	134 170		0.05

Table 6.1: Test cases and corresponding run numbers for the generation of regular waves in the wave tank of TUD.

- a matlab function *num_vs_exp.m* used to compare the measured and numerical data.

When running the comparison with experiments, two folders are created:

- *measured data*, which contains the measured wavemaker motion and velocity, as well as probe measurements after filtering; and,
- *numerical data*, which contains the results of the simulations, that is, the results presented in section 6.4.2 together with the numerical probe measurements, in text files *probe1.txt* to

probe7.txt.

6.5.3 Run a comparison with experiments of Chapter 5

A tutorial is given hereafter on how to compare the coupled tank to the measurements made in Chapter 5.

Load and filter the measured wavemaker

First, you will need to load the measured wavemaker motion and estimate its velocity. Choose a test case from the table 6.1 and run the matlab function *make_input_file.m* to create all the piston inputs. This function loads the wavemaker motion and acceleration as well as the measured depth at the probes. It filters all the measurements and estimates the wavemaker velocity from the measured wavemaker motion, using a central difference scheme:

$$u^n = \frac{\mathrm{d}x}{\mathrm{d}t} \approx \frac{x^{n+1} - x^{n-1}}{2\Delta t}, \tag{6.29}$$

where x is the measured motion and Δt the time step at which the experiments are measured. You can check that the results are consistent from the plotted figures. In particular, check that the estimated acceleration (computed from the estimated velocity) matches the measured acceleration. The piston motion, velocity, and the measured depth are saved in the folder *measured data/case 111*, if you chose to load the test case 111 for example.

Compute the simulations

Once the files are created, you can modify the settings of the coupled tank to import the data from the measurements. In the file *Settings.py*, change the *input_data* variable to "$measurements$" and set the *test_case* variable to the appropriate reference. You also need to adapt the domain dimensions to the experimental settings. Then, simply run the code *Coupling.py* to obtain the numerical data. The coupling code will interpolate the wavemaker input so that the motion and velocity are known at each solving step. The depth at the probes is saved in the folder *numerical data/case_111/124* if the test case is 111.

Compare the numerical and experimental data

The Matlab function *num_vs_exp.m* loads the experimental and numerical depth measurements, and compares their temporal evolution as well as their Fourier modes. Set the test case to the appropriate number and run the function.

6.5.4 Import other measured data

If you want to validate the code against new measurements, place the wavemaker motion $PistonMotion.dat$ and velocity $PistonVelocity.dat$ in the folder *measured data/test_case*, where "test_case" is the name set in the Settings.py file. The data will automatically be loaded through the function *load_wavemaker()* defined in the file *measured_data.py* as

```
def load_wavemaker(measurement_path):
    wm_motion = open(os.path.join(measurement_path, 'PistonMotion.dat'))
    lst=[]
    for line in wm_motion:
        lst+=[line.split()]
    wm_data = [float(x[1]) for x in lst]                  # measured motion

    wm_velocity = open(os.path.join(measurement_path, 'PistonVelocity.dat'))
    lst = []
    for line in wm_velocity:
        lst+=[line.split()]
    t_data = [float(x[0]) for x in lst]                     # measured time
    wm_vel_data = [float(x[1]) for x in lst]                      # velocity
    return wm_data, wm_vel_data, t_data
```

Then, the measured motion and velocity will automatically be interpolated by updating the time in the wavemaker expressions defined in the function *interpolate_wavemaker()*:

```
def interpolate_wavemaker(wm_data, wm_vel_data, t_data, t, dt, Lw):
    WM_expr = Expression("((wm2*(t-t1) - wm1*(t-t2))/(t2-t1))*0.5*(1.0+copysign(1.0,Lw
    -x[0]))", wm2=wm_data[1], wm1=wm_data[0], t1=t_data[0], t2=t_data[1], t=t, Lw=Lw)

    dWM_expr = Expression("((dwm2*(t-t1) - dwm1*(t-t2))/(t2-t1))*0.5*(1.0+copysign
    (1.0,Lw-x[0]))", dwm2=wm_vel_data[1], dwm1=wm_vel_data[0], t1=t_data[0], t2=t_data
    [1], t=t, Lw=Lw)
```

You might need to change the number of probes and their positions through the function *probe_location()* :

```python
def probe_location(res_dw):
    x1 = 15.002
    x2 = 17.086
    x3 = 19.040
    x4 = 20.015
    x5 = 21.084
    x6 = 22.022
    x7 = 23.159

    Ind_1 = int(x1/res_dw)
    Ind_2 = int(x2/res_dw)
    Ind_3 = int(x3/res_dw)
    Ind_4 = int(x4/res_dw)
    Ind_5 = int(x5/res_dw)
    Ind_6 = int(x6/res_dw)
    Ind_7 = int(x7/res_dw)

    return Ind_1, Ind_2, Ind_3, Ind_4, Ind_5, Ind_6, Ind_7
```

that returns the indices of the mesh coordinates corresponding to the probe locations x_1 to x_7. and consequently add or remove saving files in the functions *probe_files()* and *save_probes()*:

```python
def probe_files(save_path):
    x1_file = open(os.path.join(save_path, 'probe1.txt'), 'w')
    x2_file = open(os.path.join(save_path, 'probe2.txt'), 'w')
    x3_file = open(os.path.join(save_path, 'probe3.txt'), 'w')
    x4_file = open(os.path.join(save_path, 'probe4.txt'), 'w')
    x5_file = open(os.path.join(save_path, 'probe5.txt'), 'w')
    x6_file = open(os.path.join(save_path, 'probe6.txt'), 'w')
    x7_file = open(os.path.join(save_path, 'probe7.txt'), 'w')

def save_probes(t, h_n0, dw_beach, x1_file, x2_file, x3_file, x4_file, x5_file,
     x6_file, x7_file):
    #------------------- wave elevation probe 1 : x1 = 15m -------------------#
    x1_file.write('%-10s %-10s\n'
                  %(str(t),str(h_n0.dat.data[Ind_1]+dw_beach.dat.data[Ind_1])))
    #------------------- wave elevation probe 2 : x2 = 17m -------------------#
    x2_file.write('%-10s %-10s\n'
                  %(str(t),str(h_n0.dat.data[Ind_2]+dw_beach.dat.data[Ind_2])))
    #------------------- wave elevation probe 3 : x3 = 19m -------------------#
    x3_file.write('%-10s %-10s\n'
                    %(str(t),str(h_n0.dat.data[Ind_3]+dw_beach.dat.data[Ind_3])))
    #------------------- wave elevation probe 4 : x4 = 20m -------------------#
    x4_file.write('%-10s %-10s\n'
                    %(str(t),str(h_n0.dat.data[Ind_4]+dw_beach.dat.data[Ind_4])))
    #------------------- wave elevation probe 5 : x5 = 21m -------------------#
    x5_file.write('%-10s %-10s\n'
                    %(str(t),str(h_n0.dat.data[Ind_5]+dw_beach.dat.data[Ind_5])))
    #------------------- wave elevation probe 6 : x6 = 22m -------------------#
    x6_file.write('%-10s %-10s\n'
                    %(str(t),str(h_n0.dat.data[Ind_6]+dw_beach.dat.data[Ind_6])))
    #------------------- wave elevation probe 7 : x7 = 23m -------------------#
    x7_file.write('%-10s %-10s\n'
                    %(str(t),str(h_n0.dat.data[Ind_7]+dw_beach.dat.data[Ind_7])))
```

Finally, open each file with Matlab or Python to compare the simulated free surface to the measured one.

6.6 Conclusions

The tutorials presented in this chapter are of interest to both the maritime industry and the academic community.

Following the instructions, one can simulate waves, including extreme waves, in customized wave tanks. Henceforth, the maritime industry, such as MARIN, can test wave generation in the numerical tank at low-computational cost and good accuracy, thus improving the efficiency of experiments or large-scale simulations. Considering the high cost of experiments and the considerable computational time of large-scale simulations, reducing the number of attempts to reach the intended sea state, both experimentally and numerically, is of great interest to the maritime industry. The numerical solution obtained with our numerical tank may be used to compute a first wave-behaviour estimate, hence optimising experiments and large-scale simulations.

Our strategies to numerically capture the fourfold amplification of a solitary wave can be used for the simulation of ninefold solitary-wave amplification, by adapting the code of section 6.2 with appropriate domain characteristics and initial soliton solution, obtained from the exact solution of Baker [6]. The detailed description of our implementation strategies for the deep-water and coupled tanks will also ease the extensions to more sofisticated wave-tank modelling, such as those described in the conclusions of Chapter 3. In particular, a strategy to implement free-surface models is from now on available in section 6.3, thus facilitating enhancement of research in this field. Moreover, extension of the nonlinear coupling between the deep- and shallow-water equations with intrinsic coupling points or second-order temporal scheme, as suggested in conclusions of Chapter 4, is greatly eased by the descriptive tutorial of section 6.4.

Optimisation of the codes is facilitated by the detailed description of the code structures. In particular, the shallow-water code presented in section 6.4.4 can be optimised and adapted to be coupled within Firedrake, for example using the Firedrake library *flooddrake*, used for the implementation of finite-element shallow-water codes with C^0 discontinuous Galerkin expansions that are equivalent to the finite-volume model implemented in section 6.4.4. In addition, regular updates of Firedrake should be performed to ensure that optimised syntax is used in the codes.
Finally, by means of the detailed tutorial of section 6.5, one can access and use any measured data for validation of future models. In particular, extensions of the present numerical tank can be validated from the data collected in Chapter 5.

Chapter 7

Conclusions

7.1 Overview

As explained in Chapter 1, the design of reliable maritime structures requires the estimation of the load and stress applied on maritime devices such as ships, wind turbines, offshore platforms *etc.*. This external forcing comes mainly from waves, which have a complex structure due to the nonlinear free surface between water and air. Some particularly destructive waves, called rogue or freak waves, regularly cause accidents, sometimes with tragic consequences. These gravity waves of high amplitude appear at any depth and can neither be predicted nor avoided. Therefore, ship-design practice needs to be updated in order to build safer ships and to ensure security of crew and passengers in anticipation of a random encounter with a rogue wave.

To cater for such industrial requirements, this thesis has encompassed both the derivation and implementation of cost-effective models of water waves, including rogue waves, for diverse maritime applications. The initial maritime requirements are reviewed in section 7.2. The modelling process and achievements are then summarised in section 7.3. In order to facilitate transfer of knowledge, several outreach activities were conducted during the SurfsUp project. A summary of the main activities is given in section 7.5. Finally, prospective improvements of the present models are highlighted in section 7.4 to facilitate future extensions of this thesis.

7.2 Summary of thesis objectives

The update of maritime design practice requires knowledge about rogue-wave dynamics and statistics that is currently unavailable. Field measurements, experiments and large-scale simulations of rogue waves are moreover too costly to be used as frequently as required in maritime engineering practice. The aim of this thesis was therefore to develop a cost-effective water-wave-simulation tool for the optimisation or substitution of large-scale simulations and experimental measurements. For that purpose, the models had to meet the following specifications:

- simulations must necessitate minimal computational resources and be faster than usual large-scale simulations used at MARIN to simulate wave dynamics with the Reynolds-averaged Navier-Stokes equations (PARNASSOS, ReFRESCO...);
- numerical integrators must ensure conservation properties, including conservation of energy, momentum and mass;
- extreme physics of rogue waves must be stably captured in both deep and shallow water;
- water waves must be simulated in a numerical tank with a wavemaker, seabed topography and a damping beach;
- accuracy of the models must be validated against experimental measurements.

Advanced mathematical and numerical methods were derived to tackle the aforementioned challenges. The modelling process, achievements and direct applications are summarised in the next section.

7.3 Achievements and applications

The industrial specifications of section 7.2 were met in four steps. First, nonlinear solvers were derived to model rogue waves in shallow water. Second, dispersion was considered to model a rogue wave in a deep-water tank with wavemaker and seabed topography. Third, the deep-water model was coupled to a shallow-water beach in order to absorb the waves and reduce the computational domain for improved cost-efficiency. Finally, experimental validation was conducted to ensure accuracy of the models and facilitate future extensions.

Rogue wave in shallow water: the Benney-Luke-type model

In Chapter 2, a first step towards the modelling of extreme waves was carried out. Using a simplified model – shallow-water, weakly dispersive and weakly nonlinear – the numerical methods (namely, the finite-element discretization and the 2^{nd}–order Störmer-Verlet scheme) were robustly tested for the simulation of extreme solitary waves. The theoretical soliton reflection as predicted by Miles [105, 103] was extended and applied to our Benney-Luke-type system of equations, for which an analytical soliton solution was derived by extension of the KP-soliton initially introduced by Kodama *et al.* [84]. A comparison between our numerical simulations and the predicted soliton behaviour showed good agreement for various initial wave profiles, from regular to Mach reflection. In particular, the model was able to capture the dynamic amplification of the initial solitary wave up to 3.6, which is the highest wave amplification captured to date in the literature. The results from these simulations confirmed the efficiency, stability and accuracy of the numerical scheme in the modelling of extreme, shallow-water rogue-type waves, before proceeding to consider the extension to fully nonlinear and dispersive systems in deep water. In addition, the Benney-Luke-type model, whose implementation is explained in Chapter 6, will help the maritime industry to study the dynamics of rogue-wave formation in shallow-water crossing seas, where several accidents have been reported [109].

Extension to nonlinear and dispersive waves: the 3D deep-water tank

The numerical methods were then extended to solve the potential-flow equations for the simulation of higher, steeper nonlinear dispersive waves in a deep-water tank with seabed topography and a wavemaker. Modelling and numerical strategies were derived in Chapter 3, not only to handle moving boundaries at the wavemaker and at the nonlinear free surface but also to capture the evolution of the fluid velocity both at the dynamic surface and in depth. To guarantee cost-efficiency of the computations, optimisation of the solvers was also investigated. Consistency of the simulations was checked via a test of spatial convergence and verification of numerical-energy conservation. Accuracy of the model was verified with a validation of the simulations against an experimental focussed wave, a phenomenon that combines both nonlinear and dispersive effects. To facilitate its utilisation by the maritime industry and its extension in the research field, the model was also built to be flexible in terms of wavemaker motion, seabed profile, temporal schemes and spatial interpolations, and a tutorial was provided in Chapter 6. Thus, an efficient solution to the modelling of nonlinear dispersive waves, including extreme, rogue waves, in a wave tank similar to those used at MARIN has been achieved. The strategies introduced for the three-dimensional modelling of a fluid with a free surface can be used for many applications and in particular as a first step in the modelling of wave-structure interactions. However, reflection of the waves against the wall opposite to the wavemaker disturbs the target area in which wave-structure interactions in real-sea conditions are tested. The next step was therefore to couple this deep-water model to an absorbing-wave model in order to reduce wave reflection without increasing the length of the computational domain.

Absorbing breaking waves with the addition of a beach: the numerical wave tank

Coupling the deep-water model to a shallow-water beach, as done in Chapter 4, had several objectives. The first aim was to extend the numerical tank so that it matched entirely the experimental wave tank installed at MARIN. The second objective was to enable the simulation and absorption of waves approaching the breaking-wave limit, which was not possible with the potential-flow model of Chapter 3. Last, but not least, the goal was to reduce disturbance of the target area caused by reflected waves while still retaining the computational-cost efficiency obtained in the deep-water potential-flow model. In the shallow-water domain, breaking waves were stably modelled as hydraulic bores discretised with the finite-volume method. The method of Audusse [5] for capturing the moving waterline enabled stable simulations of the wet/dry beach to be computed and ensured the non-negativity of water depth. The main challenge, which was to derive and implement a stable and accurate coupling between the deep- (potential-flow) and shallow-water equations, was achieved by means of a variational coupling strategy resulting in the first fully nonlinear coupling able to capture steep, nonlinear waves. Results showed consistent, continuous and stable transfer of energy through the coupling interface from deep to shallow water and *vice versa*. In addition, the ability of the numerical model to absorb wave energy was shown to be efficient, thus optimising the computational time by keeping a relatively short computational domain.

Experimental validation of the numerical wave tank

To be operable for maritime applications, accuracy of the coupling had to be verified. Experimental measurements were conducted at the Delft University of Technology to validate the accuracy and test the limits of the numerical tank. Four main features were tested: the ability to capture and absorb irregular waves; the accuracy in absorbing long waves; the ability to capture amplitude modulation of short waves resulting from the piston motion; and, the stability, continuity and accuracy of the coupling process in the case of steep waves. Comparisons in the deep-water part showed good agreement for all types of waves, thus confirming the accuracy of the deep-water potential-flow model. In addition, the free-surface depth at the coupling point was stable and continuous in all four cases, thus confirming the efficiency of the variational-coupling process in transferring information from deep to shallow and from shallow to deep water. Finally, the validation showed that the beach efficiently absorbed incoming waves, but that its accuracy can be improved, in particular for short waves that require a shallow coupling interface. Overall, the experimental validation of the numerical tank has made the model of interest and useful to both the maritime industry and the research community. By matching the design of the computational domain to MARIN's experimental tank, the numerical tank can be used by MARIN to test waves before generating them in their basins. For example, by using the numerical wave tank, MARIN can account for amplitude modulation resulting from the translational motion of the piston wavemaker to adjust the experimental wavemaker input and correct the disturbance of the wave amplitudes. Similarly, for the testing of wave impact upon a vessel or a wind turbine, the wavemaker can be tuned to generate a rogue wave in a target area of the experimental basin. Considering the price of model tests, reducing the number of trials before obtaining the requested wave profile is indeed of interest for MARIN and attractive for their clients. Moreover, repeated measurements of wave impact upon maritime structures required for the update of design practice can be conducted in my simulations at low cost. A comparison with the large-scale simulations used at MARIN (ReFRESCO and OceanWave3D) shows that their fastest solver requires 21 hours to compute the focused wave presented in Chapter 3, while our program is able to accurately capture the focussed wave in one to two hours using the same number (*i.e.* one) of cores. This considerable economy of time, energy and therefore money is of great interest to MARIN, which will use the results of our simulations as either a substitute for, or a first guess of, their large-scale simulations. The use and extension of the numerical tank is also facilitated by the detailed code tutorials provided in Chapter 6, which effectively comprise a manual for future users.

7.4 Extensions of the present numerical models

In this section, possible improvements of the present numerical models are summarised in order to facilitate future work in the field of water-wave modelling.

7.4.1 Extensions of the shallow-water rogue-wave model

The numerical methods used to derive and discretise the shallow-water model presented in Chapter 2 have yielded the simulation of the highest dynamic amplification obtained in the literature. While this result confirms the efficiency of our numerical scheme for capturing extreme waves, some improvements could enable one to get even closer to the fourfold amplification predicted by Miles [105, 103].

Due to limited computational resources, the spatial resolution used to simulate soliton interactions with our Benney-Luke model is not accurate enough to allow the simulation of the fourfold amplification. The transition from regular to Mach reflection being extremely abrupt, one must ensure that a sufficiently high resolution is used to capture the maximal amplification. The large computational domain relative to the soliton length implies that thousands of nodes should be considered. Inhomogeneous meshing and parallel programming are therefore two solutions for reducing the computational cost. In addition, an optimisation of the solvers as performed for the potential-flow solver in Chapter 3 could improve the computational efficiency of the Benney-Luke model.

Moreover, higher amplitude waves could be simulated using the interaction of three solitons instead of two. To this end, the initial KP-soliton wave pattern derived by Baker [6] can be transformed into a solution of our Benney-Luke model following the method given in Chapter 2. Running the shallow-water model with this initial wave pattern should yield the expected eight-to-ninefold amplified stem wave.

Finally, the initial soliton profile used to simulate the fourfold amplification with the Benney-Luke model could be transformed into a solution of the potential-flow equations, using the transformations introduced in Chapter 2 to obtain the Benney-Luke simplification. As the potential-flow model is more accurate than the simplified Benney-Luke model for representing water waves, it will be interesting to assess the level of agreement between the numerical amplification and experimental measurements or theory.

7.4.2 Extensions of the deep-water tank

The potential-flow model derived and implemented in Chapter 3 was shown to be both accurate and cost-effective for the simulation of rogue-type waves. However, while the piston wavemaker implemented for generating the waves is useful to predict amplitude modulation of short waves in experimental tank, its extension to a flap-type wavemaker would allow more accurate simulations of realistic sea states and, in particular, of deep-water waves. In terms of experimental applications, the addition of a second wavemaker on the tangent wall would make it possible to simulate wave propagation from several directions. As a result, wave-wave interaction could be simulated to test stronger wave impact on structures. Finally, defining the target area from the simulations would be eased by solving inverse problems that compute the transfer function of the wavemaker input relative to the expected location of the rogue-type wave. This inverse problem concerns both the simulation of rogue-type waves in one direction (*e.g.,* by using the dispersion effect to obtain a focussed wave) or the interaction of multidirectional waves.

7.4.3 Extensions of the coupling process

The coupling process developed in Chapter 4 and validated in Chapter 5 was shown to be both efficient and accurate for continuously transferring information from deep to shallow water and from shallow to deep water. However, several improvements were also highlighted.

First, the temporal scheme implemented in Chapter 4 is a simplification of the symplectic-Euler scheme used to optimise the computational time. The proper symplectic-Euler scheme is, however, presented in Chapter 4 and can be implemented using a fully-implicit evaluation of the water depth, which would increase the accuracy of the temporal scheme but also require additional iterations.

Following the same method, the second-order Störmer-Verlet scheme can be implemented. A performance comparison of these schemes would improve the flexibility of the numerical tank by giving more options to the user depending on his objective. As explained in Chapter 3, the choice between a first- and second-order scheme must be made depending on both the accuracy and computational speed required.
Second, extension of the coupling process to 3D would yield a fully 3D numerical tank, since the 3D potential-flow and surface shallow-water models have already been developed. In addition, extension to an intrinsic location of the coupling interface would avoid the compromise between deep-water stability and shallow-water accuracy highlighted in Chapter 5 for short waves. These improvements would therefore broaden the application spectrum of the numerical tank.

7.4.4 Extensions of the shallow-water absorbing beach

The finite-volume shallow-water beach implemented for absorbing waves is an efficient way to capture breaking waves, which was not possible with the continuous finite-element method used to discretise the potential-flow model. However, its implementation, as detailed in Chapter 6, and computational speed could be improved by means of an optimisation process and parallel programming. In addition, the implementation of a finite-element solver with discontinuous Galerkin elements, using for instance the Firedrake library Flooddrake, would facilitate the intrinsic coupling of the deep- and shallow-water models since they would both be solved within Firedrake. To test breaking-wave impact on structures, the hydraulic-bore model can be extended to more realistic breaking-wave models that can then be validated with the experimental record of the waterline and breaking location mentioned in Chapter 6.

7.5 Outreach activities

To increase people's interest for mathematics and enhance research in the field, several outreach activities were conducted during the project. Public fluid-wave-tank demonstrations were given, at the University of Leeds Open Days, using the small-scale experimental set up designed by W. Booker, J. van Alwon and T. Goodfellow [21]. The tank consists of a water channel with a motor-driven flap-type wavemaker on one side and vertical walls on the other boundaries. Several examples of fluid-dynamics modelling applications were presented to a great number of interested visitors; examples included the generation of standing waves, breaking waves on a beach, waves impacting upon a wind-turbine model and the effect of the coastal design in wave reflection (*cf.* Fig. 7.1). Water-wave simulations were shown and compared to the live demonstrations, which had generated great public interest and attention, and convinced many students (and their parents) of the importance of mathematics for solving real-world challenges.

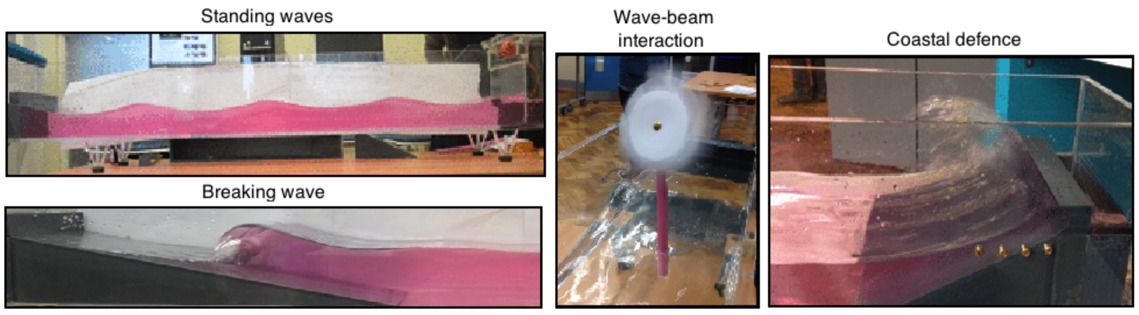


Figure 7.1: Examples of demonstrations given at Leeds open days. More details are given in the SurfsUp blog.

A lecture on the evolution of beliefs and knowledge about freak waves was also given in the programme of regular talks at the well-known and well-established Café Scientifique in Leeds. The predominantly non-scientific audience was fascinated by the story of these mysterious waves. Our discourses were summarised in our blog (https://blogsurfsup.wordpress.com) and Facebook page (https://www.facebook.com/surfsupeueid/), together with additional articles about our progress (new simulations, experiments *etc.*). The blog was visited hundreds of times and from more than 30 countries (*cf.* blog statistics in Fig. 7.2).

In addition, my presentation at the Leeds Doctoral Showcase 2017 received great interest both from the panel and the lay audience, and was rewarded by the prize of *Postgraduate Researcher of the Year 2017* (*cf.* Fig. 7.3). This award considerably increased the outreach activity about the SurfsUp project since it was shared in social media and news articles were published in the University of Leeds website, by MARIN on their website and by the reputable French engineering school INSA Toulouse. An example of this successful outreach communication is the Youtube video of my presentation (https://youtu.be/6gKcWKeZ5Xs) that has been watched by almost 800 people to this day. In addition, I was named "early-stage researcher of the week" (January 2018) by the Marie Curie actions; their post about my research and the SurfsUp project was shared more than 120 times on social media, attesting interest and further widening the impact of the research.

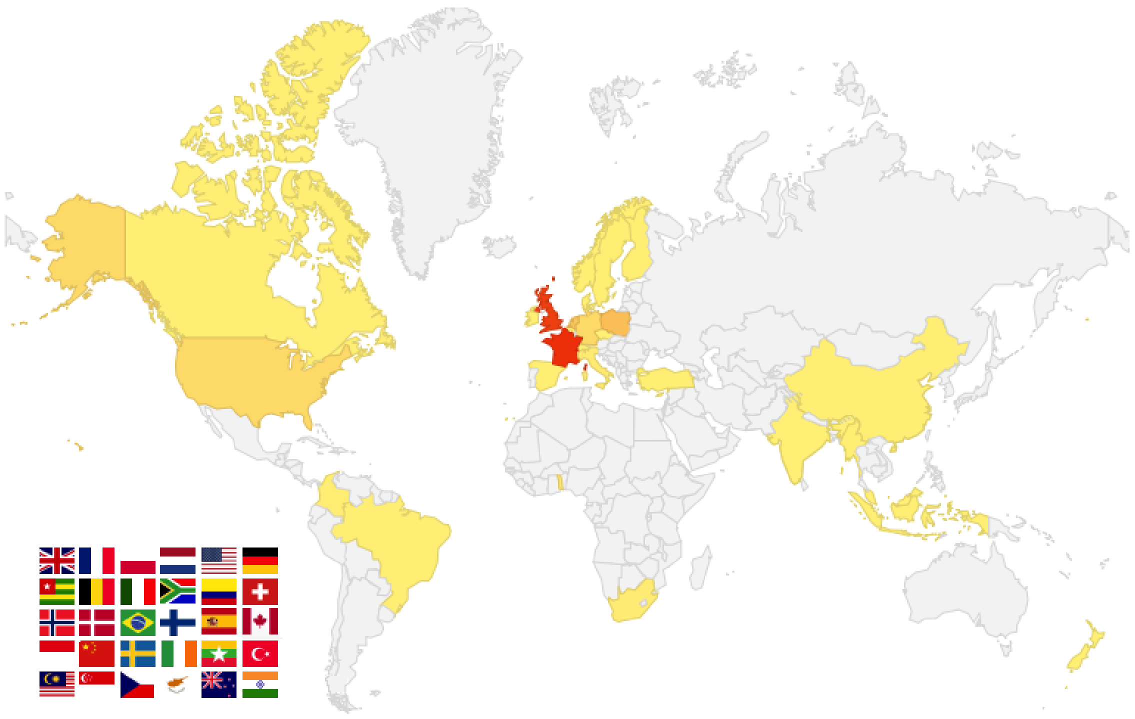


Figure 7.2: Geographical origin of visitors to our blog: darker colours correspond to a higher frequency of visits.

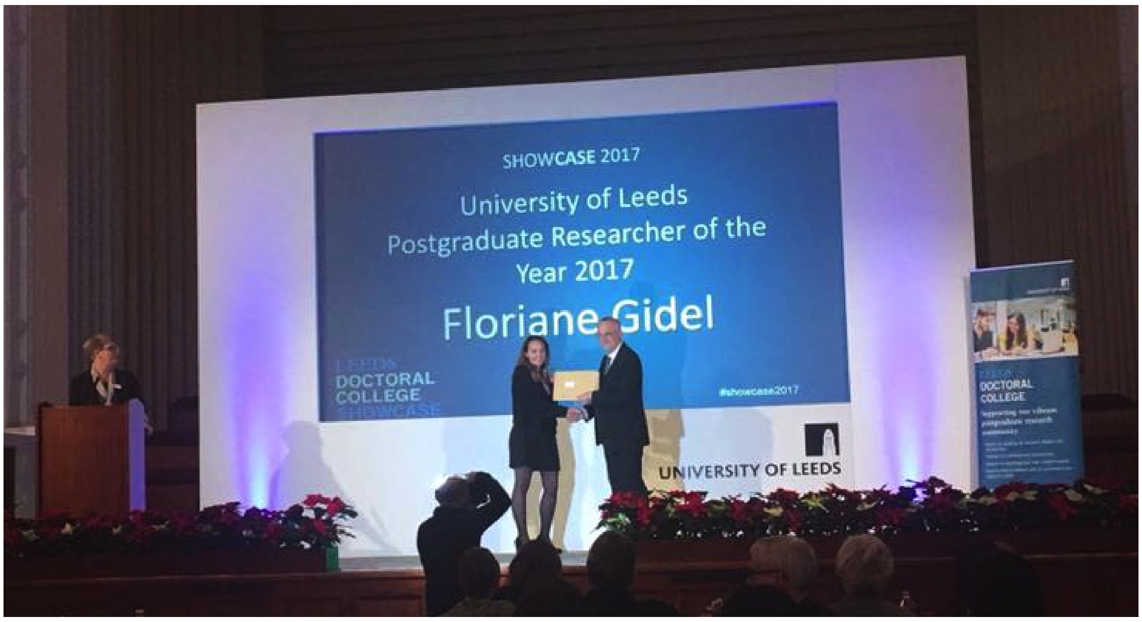


Figure 7.3: Award of Postgraduate Researcher of the year 2017 at the Leeds Doctoral Showcase. Photo courtesy: Arththi Paathi.

Appendices

A Rogue-type waves in shallow water: the example of solitary-wave interactions

A.1 Time discretization of the Benney–Luke model

The Störmer–Verlet scheme (2.61) is applied to the variational principle (2.51) for Benney–Luke, with $\mathbf{Q} = \left\{\left(\mathbf{M}_{ij} + \frac{\mu}{2}\mathbf{A}_{ij}\right)\eta_i\right\}$ and $\mathbf{P} = \{\Phi_i\}$, leading to

$$\begin{aligned}0 = \int_{\Omega_\mathrm{b}} &\left(\Phi_i^{n+1/2} - \Phi_i^n\right)\left[\varphi_\mathrm{i}\varphi_j + \frac{\mu}{2}\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_j\right]\\ &+ \frac{\Delta t}{2}\left[\eta_j^n\varphi_j\varphi_k + \frac{\epsilon}{2}\varphi_j\Phi_i^{n+1/2}\Phi_k^{n+1/2}\nabla\varphi_\mathrm{i}.\cdot\nabla\varphi_k\right]\mathrm{d}\Omega_\mathrm{b},\end{aligned} \tag{A.7a}$$

$$0 = \int_{\Omega_\mathrm{b}}\left(q_i^{n+1/2}\varphi_\mathrm{i}\varphi_j - \frac{2}{3}\nabla\varphi_j\cdot\nabla\varphi_\mathrm{i}\Phi_i^{n+1/2}\right)\mathrm{d}\Omega_\mathrm{b}, \tag{A.7b}$$

$$\begin{aligned}0 = \int_{\Omega_\mathrm{b}} &\left(\eta_i^{n+1}\varphi_\mathrm{i} - \eta_j^n\varphi_j\right)\varphi_k + \frac{\mu}{2}\nabla\varphi_k\cdot\nabla\left(\eta_i^{n+1}\varphi_\mathrm{i} - \eta_j^n\varphi_j\right)\\ &- \frac{\Delta t}{2}\Bigg[\left((1+\epsilon\eta_i^n\varphi_\mathrm{i})\nabla\varphi_k\cdot\nabla\varphi_j\Phi_j^{n+1/2} - \mu q_i^{n+1/2}\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_k\right)\\ &+ \left((1+\epsilon\eta_i^{n+1}\varphi_\mathrm{i})\nabla\varphi_k\cdot\nabla\varphi_j\Phi_j^{n+1/2} - \mu q_i^{n+1/2}\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_k\right)\Bigg]\mathrm{d}\Omega_\mathrm{b},\end{aligned} \tag{A.7c}$$

$$\begin{aligned}0 = \int_{\Omega_\mathrm{b}} &\left(\Phi_i^{n+1} - \Phi_i^{n+1}\right)\left[\varphi_\mathrm{i}\varphi_j + \frac{\mu}{2}\nabla\varphi_\mathrm{i}\cdot\nabla\varphi_j\right]\\ &+ \frac{\Delta t}{2}\left[\eta_j^{n+1}\varphi_j\varphi_k + \frac{\epsilon}{2}\varphi_j\Phi_i^{n+1/2}\Phi_k^{n+1/2}\nabla\varphi_\mathrm{i}.\cdot\nabla\varphi_k\right]\mathrm{d}\Omega_\mathrm{b}.\end{aligned} \tag{A.7d}$$

B Rogue-type waves in a deep-water tank

B.1 Transformed equations

In this section, we show how to obtain the transformed Euler equations from the transformed variational principle. In the original system of coordinates, the Euler equations are obtained from Luke's variational principle (3.4) and consist of the Laplace equation augmented by a kinematic and a dynamic boundary conditions at the free surface:

$$\nabla^2\phi = 0, \qquad \text{in } \Omega, \tag{3.3a}$$

$$\partial_t h + \nabla h \cdot \nabla\phi - \partial_z\phi = 0, \qquad \text{at } z = h, \tag{3.3b}$$

$$\partial_t\phi + \frac{1}{2}|\nabla\phi|^2 + g(h - H) = 0, \qquad \text{at } z = h, \tag{3.3c}$$

$$\partial_x\phi - \partial_y\phi\partial_y R = \partial_t R \qquad \text{at } x = R. \tag{3.3d}$$

Equation (3.3a) is the Laplace equation, while Eqn. (3.3b) and (3.3c) are the conservation of mass and momentum boundary conditions respectively. Equation (3.3d) sets the velocity on the left boundary to be equal to the wavemaker velocity. On the other walls, Neumann boundary conditions are applied. Equations (3.3) may be transformed with scaling (3.6), respectively leading to:

$$\begin{aligned}
0 = &\left[\partial_{yy}\phi - \frac{z}{h}\left(\partial_{yy}h\partial_z\phi + 2\partial_y h\partial_{yz}\phi\right)\right.\\
&\quad\left. + 2\frac{z}{h^2}(\partial_y h)^2\partial_z\phi + \frac{z^2}{h^2}(\partial_y h)^2\partial_{zz}\phi + \frac{H_0^2}{h^2}\partial_{zz}\phi\right]\\
&+\left[\partial_{xx}\phi - \frac{z}{h}\left(\partial_{xx}h\partial_z\phi + 2\partial_x h\partial_{xz}\phi\right)\right.\\
&\quad\left. + 2\frac{z}{h^2}(\partial_x h)^2\partial_z\phi + \frac{z^2}{h^2}(\partial_x h)^2\partial_{zz}\phi\right]\left[\frac{L_w^2}{W^2} + \frac{U^2}{W^2}\right]\\
&+\left[\partial_x\phi - \frac{z}{h}\partial_x h\partial_z\phi\right]\left[\frac{2U}{W^2}\partial_y\tilde{R} + \frac{x - L_w}{W}\partial_{yy}\tilde{R}\right]\\
&+\left[\partial_{xy}\phi - \frac{z}{h}\left(\partial_{xy}h\partial_z\phi + \partial_x h\partial_{yz}\phi + \partial_y h\partial_{xz}\phi\right)\right.\\
&\quad\left. + 2\frac{z}{h^2}\partial_x h\partial_y h\partial_z\phi + \frac{z^2}{h^2}\partial_x h\partial_y h\partial_{zz}\phi\right]\frac{2U}{W}, \qquad \text{in } \hat{\Omega},
\end{aligned} \tag{B.9a}$$

$$\begin{aligned}
0 =&\partial_t h + \partial_y h \partial_y \phi - \partial_z \phi \frac{H_0}{h}\left(1 + (\partial_y h)^2\right) + \frac{(x - L_w)}{W}\partial_t \tilde{R} \partial_x h \\
&+ \frac{U}{W}\left[\partial_y h \partial_x \phi + \partial_x h \partial_y \phi - 2\frac{H_0}{h}\partial_x h \partial_y h \partial_z \phi\right] \\
&+ \frac{V}{W^2}\left[\partial_x h \partial_x \phi - \frac{H_0}{h}(\partial_x h)^2 \partial_z \phi\right] \qquad \text{at } z = H_0,
\end{aligned} \tag{B.9b}$$

$$\begin{aligned}
0 =&\partial_t \phi + \frac{1}{2}(\partial_y \phi)^2 + g(h - H) + \frac{1}{2}\frac{H_0^2}{h^2}(\partial_z \phi)^2\left(1 + (\partial_y h)^2\right) \\
&- \frac{H_0}{h}\left[\partial_t h \partial_z \phi + \partial_y h \partial_y \phi \partial_z \phi\right] + \frac{(x - L_w)}{W}\partial_t \tilde{R}\left[\partial_x \phi - \frac{H_0}{h}\partial_x h \partial_z \phi\right] \\
&+ \frac{U}{W}\left[\partial_x \phi \partial_y \phi + \frac{H_0^2}{h^2}\partial_x h \partial_y h (\partial_z \phi)^2 - \frac{H_0}{h}\Big(\partial_x h \partial_y \phi \partial_z \phi + \partial_y h \partial_x \phi \partial_z \phi\Big)\right] \\
&+ \frac{V}{2W^2}\left[(\partial_x \phi)^2 + \frac{H_0^2}{h^2}(\partial_x h)^2(\partial_z \phi)^2 - 2\frac{H_0}{h}\partial_x h \partial_x \phi \partial_z \phi\right] \qquad \text{at } z = H_0,
\end{aligned} \tag{B.9c}$$

$$\partial_t \tilde{R} = \frac{L_w + L_w(\partial_y \tilde{R})^2}{W}\Big(\partial_x \phi - \frac{z}{h}\partial_x h \partial_z \phi\Big) - \partial_y \tilde{R}\Big(\partial_y \phi - \frac{z}{h}\partial_y h \partial_z \phi\Big), \text{ at } x = 0, \tag{B.9d}$$

in which we have used that

$$\begin{aligned}
\partial_{xx}\phi \to& \frac{L_w^2}{W^2}\left[\partial_{xx}\phi - \frac{z}{h}\left(\partial_{xx}h \partial_z \phi + 2\partial_x h \partial_{xz}\phi\right) + \frac{z}{h^2}(\partial_x h)^2\left(2\partial_z \phi + z\partial_{zz}\phi\right)\right], \\
\partial_{yy}\phi \to& \frac{U^2}{W^2}\left[\partial_{xx}\phi - \frac{z}{h}\left(\partial_{xx}h \partial_z \phi + 2\partial_x h \partial_{xz}\phi\right) + \frac{z}{h^2}(\partial_x h)^2\left(2\partial_z \phi + z\partial_{zz}\phi\right)\right] \\
&+ \left[\frac{2U}{W^2}(\partial_y \tilde{R}) + \frac{(x - L_w)}{W}\partial_{yy}\tilde{R}\right]\left[\partial_x \phi - \frac{z}{h}\partial_x h \partial_z \phi\right] \\
&+ \frac{2U}{W}\left[\partial_{xy}\phi - \frac{z}{h}\left(\partial_{xy}h \partial_z \phi + \partial_x h \partial_{yz}\phi + \partial_y h \partial_{xz}\phi\right) + \frac{z}{h^2}\partial_x h \partial_y h\left(2\partial_z \phi + z\partial_{zz}\phi\right)\right] \\
&+ \left[\partial_{yy}\phi - \frac{z}{h}\left(\partial_{yy}h \partial_z \phi + 2\partial_y h \partial_{yz}\phi\right) + \frac{z}{h^2}(\partial_y h)^2\left(2\partial_z \phi + z\partial_{zz}\phi\right)\right], \\
\partial_{zz}\phi \to& \frac{H_0^2}{h^2}\partial_{zz}\phi.
\end{aligned}$$

Equation (B.9b) may be substituted into Eqn. (B.9c) so that the transformed momentum equation becomes:

$$\begin{aligned} 0 =&\partial_t\phi + \frac{1}{2}(\partial_y\phi)^2 + g(h-H) - \frac{1}{2}\frac{H_0^2}{h^2}(\partial_z\phi)^2(1+(\partial_y h)^2) + \frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x\phi \\ &+ \frac{U}{W}\left[\partial_x\phi\partial_y\phi - \frac{H_0^2}{h^2}\partial_x h\partial_y h(\partial_z\phi)^2\right] + \frac{V}{2W^2}\left[(\partial_x\phi)^2 - \frac{H_0^2}{h^2}(\partial_x h)^2(\partial_z\phi)^2\right]. \end{aligned} \tag{B.11}$$

We now show that the same equations may be obtained from the transformed variational principle Eqn. (3.15). Taking the variations of ϕ and h in Eqn. (3.15) leads to

$$\begin{aligned} \int_0^T \int_{\hat{\Omega}} \Bigg[&\frac{V}{2W}\Bigg(\delta h(\partial_x\phi)^2 + 2h\partial_x\phi\partial_x\delta\phi + 2\frac{z^2}{h}\partial_x h\left(\partial_x\delta h(\partial_z\phi)^2 + \partial_x h\partial_z\phi\partial_z\delta\phi\right) \\ &\quad - \frac{z^2}{h^2}\delta h(\partial_x h)^2(\partial_z\phi)^2 - 2z\left(\partial_x\delta h\partial_x\phi\partial_z\phi + \partial_x h\partial_x\delta\phi\partial_z\phi + \partial_x h\partial_x\phi\partial_z\delta\phi\right)\Bigg) \\ &+\frac{W}{2}\Bigg(\delta h(\partial_y\phi)^2 + 2h\partial_y\phi\partial_y\delta\phi + 2\frac{z^2}{h}\partial_y h\left(\partial_y\delta h(\partial_z\phi)^2 + \partial_y h\partial_z\phi\partial_z\delta\phi\right) \\ &\quad - \frac{z^2}{h^2}\delta h(\partial_y h)^2(\partial_z\phi)^2 - 2z\left(\partial_y\delta h\partial_y\phi\partial_z\phi + \partial_y h\partial_y\delta\phi\partial_z\phi + \partial_y h\partial_y\phi\partial_z\delta\phi\right)\Bigg) \\ &+U\Bigg(\delta h\partial_x\phi\partial_y\phi + \frac{z^2}{h}\left(\partial_x\delta h\partial_y h(\partial_z\phi)^2 + \partial_x h\partial_y\delta h(\partial_z\phi)^2 + 2\partial_x h\partial_y h\partial_z\phi\partial_z\delta\phi\right) \\ &\quad + h\partial_x\delta\phi\partial_y\phi - z\partial_z\phi\left(\partial_x\delta h\partial_y\phi + \partial_x h\partial_y\delta\phi + \partial_y\delta h\partial_x\phi + \partial_y h\partial_x\delta\phi\right) \\ &\quad + h\partial_x\phi\partial_y\delta\phi - z\partial_z\delta\phi\left(\partial_x h\partial_y\phi + \partial_y h\partial_x\phi\right) - \frac{z^2}{h^2}\delta h\partial_x h\partial_y h(\partial_z\phi)^2\Bigg) \\ &+\frac{WH_0^2}{2h}\left(2\partial_z\phi\partial_z\delta\phi - \frac{1}{h}\delta h(\partial_z\phi)^2\right)\Bigg]\,\mathrm{d}z \\ +H_0\Bigg[&W\left(g\delta h(h-H) - \phi\partial_t\delta h - \partial_t h\delta\phi\right) - (x-L_w)\partial_t\tilde{R}\left(\delta\phi\partial_x h + \phi\partial_x\delta h\right)\Bigg]_{z=H_0}\mathrm{d}x\mathrm{d}y \\ +\int_0^{L_y}\int_0^{H_0} &\left[L_w\partial_t\tilde{R}\left(\phi\delta h + h\delta\phi\right)_{x=0}\right]\mathrm{d}z\,\mathrm{d}y\,\mathrm{d}t = 0. \end{aligned}$$

Integrations by part of the terms involving spatial or temporal derivatives of $\delta\phi$ and δh and arbitrariness of $\delta\phi$ and δh lead to

$$
\begin{aligned}
&\int_0^T \int_{\hat{\Omega}} \Bigg\{ \delta\phi \Bigg[-\frac{V}{W}\Big(h\partial_{xx}\phi + \frac{z}{h}(\partial_x h)^2\left(2\partial_z\phi + z\partial_{zz}\phi\right) - z\left(\partial_z\phi\partial_{xx}h + 2\partial_x h\partial_{xz}\phi\right)\Big) \\
&\qquad\qquad + \left(\frac{2U}{W}(\partial_y\tilde{R}) + (x-L_w)\partial_{yy}\tilde{R}\right)\left(z\partial_x h\partial_z\phi - h\partial_x\phi\right) - W\Big(h\partial_{yy}\phi \\
&\qquad\qquad\quad + \frac{z}{h}(\partial_y h)^2\left(2\partial_z\phi + z\partial_{zz}\phi\right) - z\left(\partial_z\phi\partial_{yy}h + 2\partial_y h\partial_{yz}\phi\right) + \frac{H_0^2}{h}\partial_{zz}\phi\Big) \\
&\qquad\quad -2U\Big(h\partial_{xy}\phi + \frac{z}{h}\partial_x h\partial_y h\left(2\partial_z\phi + z\partial_{zz}\phi\right) \\
&\qquad\qquad\quad - z\left(\partial_z\phi\partial_{xy}h + \partial_x h\partial_{yz}\phi + \partial_y h\partial_{xz}\phi\right)\Big)\Bigg]\Bigg\}\,\mathrm{d}z \\
&\quad +\delta\phi_{H_0} H_0 \Bigg[\frac{V}{W}\left(\frac{H_0}{h}(\partial_x h)^2\partial_z\phi - \partial_x h\partial_x\phi\right) - (x-L)\partial_t R\partial_x h \\
&\qquad\qquad + U\left(-\partial_x h\partial_y\phi - \partial_y h\partial_x\phi + 2\frac{H_0}{h}\partial_x h\partial_y h\partial_z\phi\right) \\
&\qquad\qquad + W\left(\frac{H_0}{h}(\partial_z\phi)\left(1 + H_0(\partial_y h)^2\right) - \left(\partial_y h\partial_y\phi + \partial_t h\right)\right)\Bigg]_{z=H_0} \\
&\qquad + \left[\delta\phi\left(\frac{WH_0^2}{h}\partial_z\phi\right)\right]_{z=0}\mathrm{d}x\,\mathrm{d}y \\
&\pm \int_0^{L_x}\int_0^{H_0}\left[\delta\phi\Big(W\left(z\partial_y h\partial_z\phi - h\partial_y\phi\right) + U\left(z\partial_x h\partial_z\phi - h\partial_x\phi\right)\Big)\right]_{y=0,L_y}\mathrm{d}x\mathrm{d}z \\
&+ \int_0^{L_y}\int_0^{H_0}\left[\delta\phi\left(\frac{V}{W}\left(z\partial_x h\partial_z\phi - h\partial_x\phi\right) + L\partial_y R\left(h\partial_y\phi - z\partial_y h\partial_z\phi\right) + L\partial_t R h\right)\right]_{x=0} \\
&\qquad + \left[\delta\phi\Big(L_w\left(h\partial_x\phi - z\partial_x h\partial_z\phi\right)\Big)\right]_{x=L_x}\mathrm{d}y\,\mathrm{d}z\mathrm{d}t = 0,
\end{aligned}
\tag{B.12}
$$

and

$$
\begin{aligned}
\int_0^T \int_{\hat{\Omega}} \Bigg\{ \delta h \Bigg[& \frac{V}{2W} \Bigg((\partial_x \phi)^2 - 2\frac{z^2}{h} \left(\partial_{xx} h (\partial_z \phi)^2 + \partial_x h \partial_x (\partial_z \phi)^2 \right) \\
& \qquad + \frac{z^2}{h^2} (\partial_x h)^2 (\partial_z \phi)^2 + z \left(\partial_z (\partial_x \phi)^2 + 2\partial_{xx}\phi \partial_z \phi \right) \Bigg) \\
& + \frac{W}{2} \Bigg((\partial_y \phi)^2 - 2\frac{z^2}{h} \left(\partial_{yy} h (\partial_z \phi)^2 + \partial_y h \partial_y (\partial_z \phi)^2 \right) \\
& \qquad + \frac{z^2}{h^2} (\partial_y h)^2 (\partial_z \phi)^2 + z \left(\partial_z (\partial_y \phi)^2 + 2\partial_{yy}\phi \partial_z \phi \right) - \frac{H_0^2}{h^2} (\partial_z \phi)^2 \Bigg) \\
& + U \Bigg(\partial_x \phi \partial_y \phi - \frac{z^2}{h} \left(2\partial_{xy} h (\partial_z \phi)^2 + \partial_x h \partial_y (\partial_z \phi)^2 + \partial_y h \partial_x (\partial_z \phi)^2 \right) \\
& \qquad + z \left(\partial_{xz}\phi \partial_y \phi + 2\partial_z \phi \partial_{xy} \phi + \partial_{yz} \phi \partial_x \phi \right) + \frac{z^2}{h^2} \partial_x h \partial_y h (\partial_z \phi)^2 \Bigg) \\
& + \left(\frac{2U}{W} (\partial_y R) + (x - L_w) \partial_{yy} R \right) \left(z \partial_x \phi \partial_z \phi - \frac{z^2}{h} \partial_x h (\partial_z \phi)^2 \right) \Bigg] \Bigg\} \, \mathrm{d}z \qquad \text{(B.13)} \\
+ \Bigg[& \delta h H_0 W \left(g(h - H) + \partial_t \phi \right) + \partial_t R (x - L) \partial_x \phi \Bigg]_{z = H_0} \\
\pm \Bigg[& \delta h \Bigg[W \left(\frac{z^2}{h} \partial_y h (\partial_z \phi)^2 - z \partial_y \phi \partial_z \phi \right) + U \left(\frac{z^2}{h} \partial_x h (\partial_z \phi)^2 - z \partial_z \phi \partial_x \phi \right) \Bigg] \Bigg]_{y = 0, L_y} \\
+ \Bigg[& \delta h \Bigg[\frac{V}{W} \left(-\frac{z^2}{h} \partial_x h (\partial_z \phi)^2 + z \partial_x \phi \partial_z \phi \right) \\
& + L_w \partial_y \tilde{R} \left(-z \partial_z \phi \partial_y \phi + \frac{z^2}{h} \partial_y h (\partial_z \phi)^2 \right) + L \partial_t R \phi \Bigg] \Bigg]_{x = 0} \\
& + \Bigg[\delta h L_w \left(2\frac{z^2}{h} \partial_x h (\partial_z \phi)^2 - 2z \partial_x \phi \partial_z \phi \right) \Bigg]_{x = L_x} \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}t = 0
\end{aligned}
$$

In Eqn. (B.12), the arbitrariness of $\delta\phi$, $\delta\phi_{H_0}$, $\delta\phi_{x=0}$, $\delta\phi_{x=L_x}$ and $\delta\phi_{y=0,L_y}$ lead to the following equations:

$$\begin{aligned}\delta\phi:&\left(\frac{2U}{W}(\partial_y\tilde{R})+(x-L_w)\partial_{yy}\tilde{R}\right)(z\partial_x h\partial_z\phi-h\partial_x\phi)\\&-\frac{V}{W}\Big(h\partial_{xx}\phi+\frac{z}{h}(\partial_x h)^2\left(2\partial_z\phi+z\partial_{zz}\phi\right)-z\left(\partial_z\phi\partial_{xx}h+2\partial_x h\partial_{xz}\phi\right)\Big)\\&-W\Big(h\partial_{yy}\phi+\frac{z}{h}(\partial_y h)^2\left(2\partial_z\phi+z\partial_{zz}\phi\right)-z\left(\partial_z\phi\partial_{yy}h+2\partial_y h\partial_{yz}\phi\right)+\frac{H_0^2}{h}\partial_{zz}\phi\Big)\\&-2U\Big(h\partial_{xy}\phi+\frac{z}{h}\partial_x h\partial_y h\left(2\partial_z\phi+z\partial_{zz}\phi\right)\\&\quad-z\left(\partial_z\phi\partial_{xy}h+\partial_x h\partial_{yz}\phi+\partial_y h\partial_{xz}\phi\right)\Big)=0,\qquad\text{in }\Omega,\end{aligned}\tag{B.14a}$$

$$\begin{aligned}\delta\phi_{z=H_0}:\ H_0\Bigg[&\frac{V}{W}\left(\frac{H_0}{h}(\partial_x h)^2\partial_z\phi-\partial_x h\partial_x\phi\right)-(x-L_w)\partial_t\tilde{R}\partial_x h\\&+W\left(\frac{H_0}{h}(\partial_z\phi)\left(1+H_0(\partial_y h)^2\right)-(\partial_y h\partial_y\phi+\partial_t h)\right)\\&+U\left(-\partial_x h\partial_y\phi-\partial_y h\partial_x\phi+2\frac{H_0}{h}\partial_x h\partial_y h\partial_z\phi\right)\Bigg]=0,\qquad\text{at }z=H_0,\end{aligned}\tag{B.14b}$$

$$\begin{aligned}\delta\phi_{x=0}:L_w&\left[\partial_y\tilde{R}\left(h\partial_y\phi-z\partial_y h\partial_z\phi\right)+\partial_t\tilde{R}h\right]\\&+\frac{V}{W}\left(z\partial_x h\partial_z\phi-h\partial_x\phi\right)=0\qquad\text{at }x=0,\end{aligned}\tag{B.14c}$$

$$\delta\phi_{x=L_x}:L_w\left(h\partial_x\phi-z\partial_x h\partial_z\phi\right)=0,\qquad\text{at }x=L_x,\tag{B.15a}$$

$$\delta\phi_{z=H_0}:\frac{WH_0^2}{h}\partial_z\phi=0,\qquad\text{at }z=0,\tag{B.14d}$$

$$\delta\phi_{y=0,L_y}:W\left(z\partial_y h\partial_z\phi-h\partial_y\phi\right)+U\left(z\partial_x h\partial_z\phi-h\partial_x\phi\right)=0,\qquad\text{at }y=0,L_y.\tag{B.14e}$$

Equation (B.14a) is indeed equivalent to the transformed Laplace equation (B.9a), while Eqn. (B.14b) is the transformed conservation of mass equation, also obtained in Eqn.(B.9b). Equations (B.14c-f) are the transformed wavemaker and Neumann boundary conditions, also obtained in Eqn.(B.9c-f). We now integrate by parts some terms of Eqn. (B.13) to lead to:

$$
\begin{aligned}
\int_0^T \int_{\hat\Omega} \Bigg\{ \delta h \Bigg[& \frac{V}{2W}\bigg(\frac{4z^2}{h^2}(\partial_x h)^2(\partial_z\phi)^2 + \frac{z^3}{h^2}(\partial_x h)^2 \partial_z(\partial_z\phi)^2 \\
&\qquad - 2\frac{z^2}{h}\left(\partial_{xx}h(\partial_z\phi)^2 + \partial_x h \partial_x(\partial_z\phi)^2\right) + 2z\partial_{xx}\phi\partial_z\phi \bigg) \\
&+\frac{W}{2}\bigg(\frac{4z^2}{h^2}(\partial_y h)^2(\partial_z\phi)^2 - 2\frac{z^2}{h}\left(\partial_{yy}h(\partial_z\phi)^2 + \partial_y h\partial_y(\partial_z\phi)^2\right) \\
&\qquad + \frac{z^3}{h^2}(\partial_y h)^2\partial_z(\partial_z\phi)^2 + z\left(2\partial_{yy}\phi\partial_z\phi + \frac{H_0^2}{h^2}\partial_z(\partial_z\phi)^2\right) \bigg) \\
&+U\bigg(-\frac{z^2}{h}\left(2\partial_{xy}h(\partial_z\phi)^2 + \partial_x h\partial_y(\partial_z\phi)^2 + \partial_y h\partial_x(\partial_z\phi)^2\right) \\
&\qquad + 2z\partial_z\phi\partial_{xy}\phi + \frac{4z^2}{h^2}\partial_x h\partial_y h(\partial_z\phi)^2 + \frac{z^3}{h^2}\partial_x h\partial_y h\partial_z(\partial_z\phi)^2 \bigg) \\
&+\left[\frac{2U}{W}(\partial_y\tilde{R}) + (x-L_w)\partial_{yy}\tilde{R}\right]\left(z\partial_x\phi\partial_z\phi - \frac{z^2}{h}\partial_x h(\partial_z\phi)^2\right)\Bigg]\Bigg\}\mathrm{d}z \\
+\delta h_{H_0} H_0 \Bigg[& W\left(g(h-H) + \partial_t\phi + \frac{1}{2}(\partial_y\phi)^2 - \frac{1}{2}\frac{H_0^2}{h^2}(\partial_z\phi)^2\left(1+(\partial_y h)^2\right)\right) \\
&+ \frac{V}{2W}\left((\partial_x\phi)^2 - \frac{H_0^2}{h^2}(\partial_x h)^2(\partial_z\phi)^2\right) + \partial_t\tilde{R}(x-L_w)\partial_x\phi \\
&+ U\left(\partial_x\phi\partial_y\phi - \frac{H_0^2}{h^2}\partial_x h\partial_y h(\partial_z\phi)^2\right)\Bigg]_{z=H_0} \\
+\delta h_{x=0}\Bigg[& \frac{V}{2W}\left(-2\frac{z^2}{h}\partial_x h(\partial_z\phi)^2 + 2z\partial_x\phi\partial_z\phi\right) \\
&+L_w\partial_y\tilde{R}\left(-z\partial_z\phi\partial_y\phi + \frac{z^2}{h}\partial_y h(\partial_z\phi)^2\right) + L_w\partial_t\tilde{R}\phi\Bigg]_{x=0} \\
\pm\delta h_{y=0,L_y}\Bigg[& W\left(\frac{z^2}{h}\partial_y h(\partial_z\phi)^2 - z\partial_y\phi\partial_z\phi\right) + U\left(\frac{z^2}{h}\partial_x h(\partial_z\phi)^2 - z\partial_z\phi\partial_x\phi\right)\Bigg]_{y=0,L_y} \\
+\delta h_{x=L_x}\Bigg[& L\left(2\frac{z^2}{h}\partial_x h(\partial_z\phi)^2 - 2z\partial_x\phi\partial_z\phi\right)\Bigg]_{x=L_x} \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t = 0
\end{aligned}
\tag{B.16}
$$

After combining equation (B.16) with Eqn.(B.14), only the terms involving δh_{H_0} remain, leading to the following equation at the free surface:

$$\begin{aligned}\delta h_{H_0}: \quad H_0 W\Bigg[&\left(g(h-H)+\partial_t\phi+\frac{1}{2}(\partial_y\phi)^2-\frac{1}{2}\frac{H_0^2}{h^2}(\partial_z\phi)^2\left(1+(\partial_y h)^2\right)\right)\\&+\frac{(x-L)}{W}\partial_t R\partial_x\phi+U\left(\partial_x\phi\partial_y\phi-\frac{H_0^2}{h^2}\partial_x h\partial_y h(\partial_z\phi)^2\right)\\&+\frac{V}{2W^2}\left((\partial_x\phi)^2-\frac{H_0^2}{h^2}(\partial_x h)^2(\partial_z\phi)^2\right)\Bigg]=0, \qquad \text{at } z=H_0,\end{aligned} \tag{B.17}$$

which is indeed the transformed conservation of momentum equation, also obtained in Eqn. (B.9c).

B.2 Fully discrete symplectic-Euler scheme

Substituting the Hamiltonian Eqn. (3.32) into Eqn. (3.56a) leads to the first step of the symplectic-Euler scheme in fully discrete form, as:

$$\begin{aligned}0=&\frac{h_k^{n+1}-h_k^n}{\Delta t}\\&-(M_{kq}^{-1})^n\Bigg\{\frac{h_l^{n+1}h_p^{n+1}}{h_r^{n+1}}\Upsilon_{rmqlp}^n\left[\tilde{S}_{11}(M_{sm}^{-1})^n p_{1s}^n+\psi_{i'm}^*\tilde{S}_{i'1}\right]\\&\qquad+\frac{H_0^2}{h_r^{n+1}}J_{rmq}^n\left[\tilde{A}_{11}(M_{sm}^{-1})^n p_{1s}^n+\psi_{i'm}^*\tilde{A}_{i'1}\right]\\&\qquad+h_r^{n+1}\Bigg[\Gamma_{rmq}^n\left(\tilde{M}_{11}(M_{sm}^{-1})^n p_{1s}^n+\tilde{M}_{1i'}\psi_{i'm}^*\right)+L_w X_{rq}^n\tilde{I}_1-H_0 N_{qr}^n\\&\qquad\quad-\left(\Gamma_{mrq}^n+\Gamma_{qrm}\right)\tilde{D}_{11}(M_{sm}^{-1})^n p_{1s}^n+\left(\Gamma_{mrq}^n\tilde{D}_{1i'}+\Gamma_{qrm}^n\tilde{D}_{i'1}\right)\psi_{i'm}\Bigg]\Bigg\}\\&-\sum_{i'}h_k^{n+1}\Bigg[\Gamma_{kqm}^n\left(\tilde{M}_{i'j'}\psi_{j'q}^*+\tilde{M}_{i'1}(M_{rq}^{-1})^n p_{1r}^n\right)+L_w X_{km}^n\tilde{I}_{i'}\\&\qquad-\Gamma_{qkm}^n\left(\tilde{D}_{i'j'}\psi_{j'q}^*+\tilde{D}_{i'1}(M_{rq}^{-1})^n p_{1r}^n\right)\\&\qquad-\Gamma_{mkq}^n\left(\tilde{D}_{j'i'}\psi_{j'q}^*+\tilde{D}_{1i'}(M_{rq}^{-1})^n p_{1r}^n\right)\Bigg]\\&+\frac{1}{h_k^{n+1}}\Bigg[H_0^2 J_{kmq}^n\left(\tilde{A}_{i'j'}\psi_{j'q}^*+\tilde{A}_{i'1}(M_{rq}^{-1})^n p_{1r}^n\right)\\&\qquad+h_l^{n+1}h_p^{n+1}\Upsilon_{kmqlp}^n\left(\tilde{S}_{i'j'}\psi_{j'q}^*+(M_{rq}^{-1})^n p_{1r}^n\right)\Bigg],\end{aligned} \tag{B.18}$$

to be solved simultaneously with

$$
\begin{aligned}
\psi^*_{i'} &= \psi_{i'}(\boldsymbol{p}_{\mathbf{1}}^n, \boldsymbol{h}^{n+1}, t^n) \\
&= \Bigg[h_k^{n+1} \Big(\Gamma^n_{kqm} \tilde{M}_{i'j'} - \Gamma^n_{qkm} \tilde{D}_{i'j'} - \Gamma^n_{mkq} \tilde{D}_{j'i'} \Big) \\
&\quad + \frac{1}{h_k^{n+1}} \Big(h_l^{n+1} h_p^{n+1} \Upsilon^n_{kmqlp} \tilde{S}_{i'j'} + H_0^2 J^n_{kmq} \tilde{A}_{i'j'} \Big) \Bigg]^{-1} \\
&\quad \times \Bigg\{ \Bigg[h_k^{n+1} \Big(\Gamma^n_{kqm} \tilde{M}_{i'1} - \Gamma^n_{qkm} \tilde{D}_{i'1} - \Gamma^n_{mkq} \tilde{D}_{1i'} + L_w X^n_{km} \tilde{I}_{i'} \Big) \\
&\quad + \frac{1}{h_k^{n+1}} \Big(h_l^{n+1} h_p^{n+1} \Upsilon^n_{kmqlp} \tilde{S}_{i'1} + H_0^2 J^n_{kmq} \tilde{A}_{i'1} \Big) \Bigg] M^{-1}_{rq} p^n_{1r} \Bigg\},
\end{aligned}
\tag{B.19}
$$

for all $i' \in [2, n_z+1]$ in order to eliminate each internal layer. That way, we update both $\boldsymbol{h}$ at time t^{n+1} and $\boldsymbol{\psi_{i'}}$, for $i' \in [2, N_z]$, at an auxiliary time t^* corresponding to $\boldsymbol{\psi^*_{i'}} = \psi_{i'}(\boldsymbol{h}^{n+1}, \boldsymbol{p}_{\mathbf{1}}^n, t^n)$. Similarly, we obtain the second step of the Symplectic-Euler scheme to update $p_{1k} = M^{-1}_{kq}\psi_{1q}$ at time t^{n+1} through

$$
\begin{aligned}
p^{n+1}_{1k} =& p^n_{1k} - \Delta t \Bigg\{ \frac{1}{2} \Lambda^n_{kqm} (M^{-1}_{rq})^n p^n_{1r} \Big[\tilde{M}_{11} (M^{-1}_{sm})^n p^n_{1s} + \tilde{M}_{1i'} \psi^*_{i'm} \Big] \\
&+ \frac{1}{2} \Lambda^n_{kmq} \psi^*_{i'm} \Big[\tilde{M}_{i'1} (M^{-1}_{rq})^n p^n_{1r} + \tilde{M}_{i'j'} \psi^*_{j'q} \Big] \\
&- (M^{-1}_{rq})^n p^n_{1r} \Gamma^n_{mkq} \Big(\tilde{D}_{11} (M^{-1}_{sm})^n p^n_{1s} + \tilde{D}_{1i'} \psi^*_{i'm} \Big) \\
&- \psi^*_{i'm} \Gamma^n_{qkm} \Big(\tilde{D}_{i'1} (M^{-1}_{rq})^n p^n_{1r} + \tilde{D}_{i'j'} \psi^*_{j'q} \Big) \\
&+ \frac{h_p^{n+1}}{2h_t^{n+1}} \Bigg[\Big(\Upsilon^n_{tmqkp} + \Upsilon^n_{tmqpk} \Big) - \frac{h_l^{n+1}}{h_k^{n+1}} \Upsilon^n_{tmqlp} \Bigg] \\
&\quad \times \Big[(M^{-1}_{rq})^n p^n_{1r} \Big(\tilde{S}_{11} (M^{-1}_{sm})^n p^n_{1s} + 2\psi^*_{i'm} \tilde{S}_{i'1} \Big) + \psi^*_{i'm} \tilde{S}_{i'j'} \psi^*_{j'q} \Big] \\
&- \frac{H_0^2}{2h_k^{n+1} h_l^{n+1}} J^n_{lmq} \Big[(M^{-1}_{rq})^n p^n_{1r} \Big(\tilde{A}_{11} (M^{-1}_{sm})^n p^n_{1s} + 2\psi^*_{i'm} \tilde{A}_{i'1} \Big) + \psi^*_{i'm} \tilde{A}_{i'j'} \psi^*_{j'q} \Big] \\
&+ H_0 \big(g(h_l^{n+1} M^n_{kl} - H I^n_k) - (M^{-1}_{rq})^n p^n_{1r} N^n_{qk} \big) \\
&+ L_w \Big[X^n_{kq} (M^{-1}_{rq})^n p^n_{1r} \tilde{I}_1 + X^n_{km} \psi^*_{i'm} \tilde{I}_{i'} \Big] \Bigg\}.
\end{aligned}
\tag{B.20}
$$

As this step is explicit for $\boldsymbol{p}_{\mathbf{1}}^{n+1}$, a linear solver is used to solve Eqn. (B.20) (*qv.* Section 3.5.1). For the purpose of visualising the velocity potential in the full 3D domain, the update of the interior velocity potential may be required. It is obtained by solving

$$\begin{aligned}
\psi_{i'}^{n+1} &= \psi_{i'}(\boldsymbol{p}_{\mathbf{1}}^{n+1}, \boldsymbol{h}^{n+1}, t^{n+1}) \\
&= \Bigg[h_k^{n+1} \left(\Gamma_{kqm}^{n+1} \tilde{M}_{i'j'} - \Gamma_{qkm}^{n+1} \tilde{D}_{i'j'} - \Gamma_{mkq}^{n+1} \tilde{D}_{j'i'} \right) \\
&\quad + \frac{1}{h_k^{n+1}} \left(h_l^{n+1} h_p^{n+1} \Upsilon_{kmqlp}^{n+1} \tilde{S}_{i'j'} + H_0^2 J_{kmq}^{n+1} \tilde{A}_{i'j'} \right) \Bigg]^{-1} \\
&\quad \times \Bigg\{ \Bigg[\frac{1}{h_k^{n+1}} \left(h_l^{n+1} h_p^{n+1} \Upsilon_{kmqlp}^{n+1} \tilde{S}_{i'1} + H_0^2 J_{kmq}^{n+1} \tilde{A}_{i'1} \right) \\
&\quad + h_k^{n+1} \left(\Gamma_{kqm}^{n+1} \tilde{M}_{i'1} - \Gamma_{qkm}^{n+1} \tilde{D}_{i'1} - \Gamma_{mkq}^{n+1} \tilde{D}_{1i'} \right) \Bigg] M_{rq}^{-1} p_{1r}^{n+1} + h_k^{n+1} L_w X_{km}^{n+1} \tilde{I}_{i'} \Bigg\}.
\end{aligned} \tag{B.21}$$

B.3 Fully discrete Störmer-Verlet scheme

Substituting the Hamiltonian Eqn. (3.32) into Eqn. (3.64a) leads to the first step of the Störmer-Verlet scheme in fully discrete form :

$$\begin{aligned}
0 = {} & p_{1k}^{n+1/2} - p_{1k}^{n} \\
& + \frac{\Delta t}{2}\Bigg\{ \frac{1}{2}\Lambda_{kqm}^{n+1/2}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left[\tilde{M}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + \tilde{M}_{1i'}\psi_{i'm}^{*}\right] \\
& \quad + \frac{1}{2}\Lambda_{kmq}^{n+1/2}\psi_{i'm}^{*}\left[\tilde{M}_{i'1}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2} + \tilde{M}_{i'j'}\psi_{j'q}^{*}\right] \\
& \quad - (M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\Gamma_{mkq}^{n+1/2}\left(\tilde{D}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + \tilde{D}_{1i'}\psi_{i'm}^{*}\right) \\
& \quad - \psi_{i'm}^{*}\Gamma_{qkm}^{n+1/2}\left(\tilde{D}_{i'1}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2} + \tilde{D}_{i'j'}\psi_{j'q}^{*}\right) \\
& \quad + \frac{h_p^n}{2h_t^n}\left[(\Upsilon_{tmqkp} + \Upsilon_{tmqpk})^{n+1/2} - \frac{h_l^n}{h_k^n}\Upsilon_{tmqlp}^{n+1/2}\right] \\
& \qquad \times \left[(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left(\tilde{S}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + 2\psi_{i'm}^{*}\tilde{S}_{i'1}\right)\right. \\
& \qquad \quad \left. + \psi_{i'm}^{*}\tilde{S}_{i'j'}\psi_{j'q}^{*}\right] \\
& - \frac{H_0^2}{2h_k^n h_l^n}J_{lmq}^{n+1/2}\left[(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left(\tilde{A}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + 2\psi_{i'm}^{*}\tilde{A}_{i'1}\right)\right. \\
& \qquad \quad \left. + \psi_{i'm}^{*}\tilde{A}_{i'j'}\psi_{j'q}^{*}\right] \\
& \quad + H_0\left(g(h_l^n M_{kl}^{n+1/2} - HI_k^{n+1/2}) - (M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}N_{qk}^{n+1/2}\right) \\
& \quad + L_w\left[X_{kq}^{n+1/2}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\tilde{I}_1 + X_{km}^{n+1/2}\psi_{i'm}^{*}\tilde{I}_{i'}\right]\Bigg\},
\end{aligned} \tag{B.22}$$

to be solved simultaneously with

$$\begin{aligned}
\psi_{i'}^{*} &= \psi_{i'}(\boldsymbol{p}_{\mathbf{1}}^{n+1/2}, \boldsymbol{h}^{n}, t^{n+1/2}) \\
&= \Bigg[h_k^n\left(\Gamma_{kqm}^{n+1/2}\tilde{M}_{i'j'} - \Gamma_{qkm}^{n+1/2}\tilde{D}_{i'j'} - \Gamma_{mkq}^{n+1/2}\tilde{D}_{j'i'}\right) \\
& \qquad + \frac{1}{h_k^n}\left(h_l^n h_p^n \Upsilon_{kmqlp}^{n+1/2}\tilde{S}_{i'j'} + H_0^2 J_{kmq}^{n+1/2}\tilde{A}_{i'j'}\right)\Bigg]^{-1} \\
& \times \Bigg\{\Bigg[h_k^n\left(\Gamma_{kqm}^{n+1/2}\tilde{M}_{i'1} - \Gamma_{qkm}^{n+1/2}\tilde{D}_{i'1} - \Gamma_{mkq}^{n+1/2}\tilde{D}_{1i'}\right) \\
& + \frac{1}{h_k^n}\left(h_l^n h_p^n \Upsilon_{kmqlp}^{n+1/2}\tilde{S}_{i'1} + H_0^2 J_{kmq}^{n+1/2}\tilde{A}_{i'1}\right)\Bigg]M_{rq}^{-1}p_{1r}^{n+1/2} + h_k^n L_w X_{km}^{n+1/2}\tilde{I}_{i'}\Bigg\}.
\end{aligned} \tag{B.23}$$

The second step, which aims to update $\boldsymbol{h}$ and $\hat{\boldsymbol{\psi}}$, is obtained by substituting the Hamiltonian (3.32) into Eqn. (3.64b). Its fully discrete form is

$$
\begin{aligned}
0 = {} & \frac{h_k^{n+1} - h_k^n}{\Delta t/2} \\
& -(M_{kq}^{-1})^{n+1/2} \Bigg\{ \Gamma_{rmq}^{n+1/2} \Big[(h_r^n + h_r^{n+1}) \tilde{M}_{11} (M_{sm}^{-1})^{n+1/2} p_{1s}^{n+1/2} + \tilde{M}_{1i'} \left(\psi_{i'm}^* h_r^n + \psi_{i'm}^{**} h_r^{n+1} \right) \Big] \\
& \qquad - \Gamma_{mrq}^{n+1/2} \Big[(h_r^n + h_r^{n+1}) \tilde{D}_{11} (M_{sm}^{-1})^{n+1/2} p_{1s}^{n+1/2} + \tilde{D}_{1i'} \left(\psi_{i'm}^* h_r^n + \psi_{i'm}^{**} h_r^{n+1} \right) \Big] \\
& \qquad - \Gamma_{qrm}^{n+1/2} \Big[(h_r^n + h_r^{n+1}) \tilde{D}_{11} (M_{sm}^{-1})^{n+1/2} p_{1s}^{n+1/2} + \tilde{D}_{i'1} \left(\psi_{i'm}^* h_r^n + \psi_{i'm}^{**} h_r^{n+1} \right) \Big] \\
& \qquad + \Upsilon_{rmqlp}^{n+1/2} \Bigg[\left(\frac{h_l^n h_p^n}{h_r^n} \psi_{i'm}^* + \frac{h_l^{n+1} h_p^{n+1}}{h_r^{n+1}} \psi_{i'm}^{**} \right) \tilde{S}_{i'1} \\
& \qquad\qquad + \left(\frac{h_l^n h_p^n}{h_r^n} + \frac{h_l^{n+1} h_p^{n+1}}{h_r^{n+1}} \right) \tilde{S}_{11} (M_{sm}^{-1})^{n+1/2} p_{1s}^{n+1/2} \Bigg] \\
& \qquad + J_{rmq}^{n+1/2} H_0^2 \Bigg[\left(\frac{1}{h_r^n} \psi_{i'm}^* + \frac{1}{h_r^{n+1}} \psi_{i'm}^{**} \right) A_{i'1} \\
& \qquad\qquad + \left(\frac{1}{h_r^n} + \frac{1}{h_r^{n+1}} \right) \tilde{A}_{11} (M_{sm}^{-1})^{n+1/2} p_{1s}^{n+1/2} \Bigg] \\
& \qquad + L_w X_{rq}^{n+1/2} \tilde{I}_1 \left(h_r^n + h_r^{n+1} \right) - H_0 N_{qr}^{n+1/2} \left(h_r^n + h_r^{n+1} \right) \Bigg\} \\
& - \sum_{i'} \Bigg\{ \Gamma_{kqm}^{n+1/2} \Big[\left(h_k^n + h_k^{n+1} \right) \tilde{M}_{i'1} (M_{rq}^{-1})^{n+1/2} p_{1r}^{n+1/2} + \tilde{M}_{i'j'} \left(\psi_{j'q}^* h_k^n + \psi_{j'q}^{**} h_k^{n+1} \right) \Big] \\
& \qquad - \Gamma_{qkm}^{n+1/2} \Big[\left(h_k^n + h_k^{n+1} \right) \tilde{D}_{i'1} (M_{rq}^{-1})^{n+1/2} p_{1r}^{n+1/2} + \tilde{D}_{i'j'} \left(\psi_{j'q}^* h_k^n + \psi_{j'q}^{**} h_k^{n+1} \right) \Big] \\
& \qquad - \Gamma_{mkq}^{n+1/2} \Big[\left(h_k^n + h_k^{n+1} \right) \tilde{D}_{1i'} (M_{rq}^{-1})^{n+1/2} p_{1r}^{n+1/2} + \tilde{D}_{j'i'} \left(\psi_{j'q}^* h_k^n + \psi_{j'q}^{**} h_k^{n+1} \right) \Big] \\
& \qquad + H_0^2 J_{kmq}^{n+1/2} \Bigg[\tilde{A}_{i'j'} \left(\frac{1}{h_k^n} \psi_{j'q}^* + \frac{1}{h_k^{n+1}} \psi_{j'q}^{**} \right) + \left(\frac{1}{h^n} + \frac{1}{h^{n+1}} \right) \tilde{A}_{i'1} (M_{rq}^{-1})^{n+1/2} p_{1r}^{n+1/2} \Bigg] \\
& \qquad + \Upsilon_{kmqlp}^{n+1/2} \Bigg[\left(\frac{h_l^n h_p^n}{h_k^n} + \frac{h_l^{n+1} h_p^{n+1}}{h_k^{n+1}} \right) (M_{rq}^{-1})^{n+1/2} p_{1r}^{n+1/2} \\
& \qquad\qquad + \tilde{S}_{i'j'} \left(\frac{h_l^n h_p^n}{h_k^n} \psi_{j'q}^* + \frac{h_l^{n+1} h_p^{n+1}}{h_k^{n+1}} \psi_{j'q}^{**} \right) \Bigg] + L_w X_{km}^{n+1/2} \tilde{I}_{i'} \left(h_k^n + h_k^{n+1} \right) \Bigg\},
\end{aligned}
\tag{B.24}
$$

to be solved simultaneously with

$$\begin{aligned}\psi_{i'}^{**} = \Bigg[& h_k^{n+1}\left(\Gamma_{kqm}^{n+1/2}\tilde{M}_{i'j'} - \Gamma_{qkm}^{n+1/2}\tilde{D}_{i'j'} - \Gamma_{mkq}^{n+1/2}\tilde{D}_{j'i'}\right) \\ & + \frac{1}{h_k^{n+1}}\left(h_l^{n+1}h_p^{n+1}\Upsilon_{kmqlp}^{n+1/2}\tilde{S}_{i'j'} + H_0^2 J_{kmq}^{n+1/2}\tilde{A}_{i'j'}\right)\Bigg]^{-1} \\ & \times \Bigg\{ \Bigg[h_k^{n+1}\left(\Gamma_{kqm}^{n+1/2}\tilde{M}_{i'1} - \Gamma_{qkm}^{n+1/2}\tilde{D}_{i'1} - \Gamma_{mkq}^{n+1/2}\tilde{D}_{1i'}\right) \\ & + \frac{1}{h_k^{n+1}}\left(h_l^{n+1}h_p^{n+1}\Upsilon_{kmqlp}^{n+1/2}\tilde{S}_{i'1} + H_0^2 J_{kmq}^{n+1/2}\tilde{A}_{i'1}\right)\Bigg] M_{rq}^{-1}p_{1r}^{n+1/2} \\ & + h_k^{n+1} L_w X_{km}^{n+1/2}\tilde{I}_{i'}\Bigg\},\end{aligned} \tag{B.25}$$

for all $i' \in [2, n_z+1]$. Finally, the last step aims to update $\boldsymbol{p_1}$ at time t^{n+1} by solving Eqn. (3.64c), that is

$$\begin{aligned} p_{1k}^{n+1} = & p_{1k}^{n+1/2} - \frac{\Delta t}{2}\Bigg\{\frac{1}{2}\Lambda_{kmq}^{n+1/2}\psi_{i'm}^{**}\left[\tilde{M}_{i'1}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2} + \tilde{M}_{i'j'}\psi_{j'q}^{**}\right] \\ & + \frac{1}{2}\Lambda_{kqm}^{n+1/2}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left[\tilde{M}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + \tilde{M}_{1i'}\psi_{i'm}^{**}\right] \\ & - (M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\Gamma_{mkq}^{n+1/2}\left(\tilde{D}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + \tilde{D}_{1i'}\psi_{i'm}^{**}\right) \\ & - \psi_{i'm}^{**}\Gamma_{qkm}^{n+1/2}\left(\tilde{D}_{i'1}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2} + \tilde{D}_{i'j'}\psi_{j'q}^{**}\right) \\ & + \frac{h_p^{n+1}}{2h_t^{n+1}}\Big[(\Upsilon_{tmqkp} + \Upsilon_{tmqpk})^{n+1/2} - \frac{h_l^{n+1}}{h_k^{n+1}}\Upsilon_{tmqlp}^{n+1/2}\Big]\Big[\psi_{i'm}^{**}\tilde{S}_{i'j'}\psi_{j'q}^{**} \\ & \qquad + (M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left(\tilde{S}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + 2\psi_{i'm}^{**}\tilde{S}_{i'1}\right)\Big] \\ & - \frac{H_0^2}{2h_k^{n+1}h_l^{n+1}}J_{lmq}^{n+1/2}\Big[(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\left(\tilde{A}_{11}(M_{sm}^{-1})^{n+1/2}p_{1s}^{n+1/2} + 2\psi_{i'm}^{**}\tilde{A}_{i'1}\right) \\ & \qquad + \psi_{i'm}^{**}\tilde{A}_{i'j'}\psi_{j'q}^{**}\Big] \\ & + H_0\left(g(h_l^{n+1}M_{kl}^{n+1/2} - HI_k^{n+1/2}) - (M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}N_{qk}^{n+1/2}\right) \\ & + L_w\left[X_{kq}^{n+1/2}(M_{rq}^{-1})^{n+1/2}p_{1r}^{n+1/2}\tilde{I}_1 + X_{km}^{n+1/2}\psi_{i'm}^{**}\tilde{I}_{i'}\right]\Bigg\},\end{aligned} \tag{B.26}$$

which is explicit for $\boldsymbol{p_1}^{n+1}$. As in the case of the Symplectic-Euler scheme, $\hat{\boldsymbol{\psi}}$ may be computed at time t^{n+1} by solving Eqn. B.36.

B.4 Stability criteria for the symplectic-Euler scheme

In section 3.4.4, we derived the characteristic polynomial corresponding to the symplectic-Euler scheme as

$$\lambda^2 + (\omega^2 \Delta t^2 - 2) + 1 = 0. \tag{3.70}$$

The solution λ of the polynomial (3.70) depends on the sign of

$$\Delta = (\omega^2 \Delta t^2 - 2)^2 - 4. \tag{B.27}$$

If $\Delta > 0$, then the two possible solutions λ_+ and λ_- are

$$\lambda_+ = \frac{-(\omega^2 \Delta t^2 - 2) + \sqrt{\Delta}}{4}, \tag{B.28a}$$

$$\lambda_- = \frac{-(\omega^2 \Delta t^2 - 2) - \sqrt{\Delta}}{4}, \tag{B.28b}$$

The modulus of λ_- is

$$|\lambda_-| = \frac{1}{2}\sqrt{2\Delta + 4 + 2\sqrt{\Delta}(\omega^2 \Delta t^2 - 2)}. \tag{B.29}$$

However,

$$\begin{aligned}
\Delta > 0 &\Rightarrow \Delta t^2 \omega^2 - 2 > 0 \\
&\Rightarrow 2\sqrt{\Delta}(\Delta t^2 \omega^2 - 2) > 0 \\
&\Rightarrow \sqrt{2D + 4 + 2\sqrt{\Delta}(\Delta t^2 \omega^2 - 2)} > 2 \\
&\Rightarrow |\lambda_-| > 1.
\end{aligned}$$

Therefore, if $\Delta > 0$, *i.e* if $\Delta t > \frac{\omega}{2}$, there is at least one solution for which the symplectic-Euler scheme would be unstable. Therefore, the case $\Delta t > \frac{\omega}{2}$ does not ensure stability of the temporal scheme.

Let's now consider the case $\Delta \leq 0$, which is satisfied when $\Delta t \leq \frac{2}{\omega}$. In that case, the two possible solutions λ_+ and λ_- of (3.70) are

$$\lambda_\pm = \frac{-(\omega^2 \Delta t^2 - 2) \pm i\sqrt{\Delta}}{4}. \tag{B.30}$$

In both cases, the modulus of λ is

$$|\lambda_\pm| = \frac{1}{2}\sqrt{(\omega^2 \Delta t^2 - 2)^2 + \Delta} = \frac{1}{2}\sqrt{2\Delta + 4} \leq 1 \tag{B.31}$$

from the assumption $\Delta \leq 0$. Therefore, the scheme is stable if and only if $\Delta \leq 0$, that is

$$\Delta t \leq \frac{2}{\omega}. \tag{B.32}$$

B.5 Space-continuous-time-discrete weak formulations with the Symplectic-Euler scheme

Transforming the weak formulations (B.18), (B.19) and (B.20) in terms of $h(x,y,t)$, $\psi_1(x,y,t)$ and $\psi_{i'}(x,y,t)$ and substituting (3.91) and (3.92) yield the space-continuous time-discrete weak formulations in vectorial form, as implemented with Firedrake. The first step is to solve

$$\begin{aligned}
&\int_{\hat{\Omega}_{x,y}} W^n \varphi_q h^{n+1}\,\mathrm{d}x\,\mathrm{d}y = \int_{\hat{\Omega}_{x,y}} W^n \varphi_q h^n \\
&+\Delta t \Bigg\{ h^{n+1}\left[\frac{V^n}{W^n}\partial_x\varphi_q + U^n\partial_y\varphi_q\right]\left[\partial_x\psi_1^n\tilde{M}_{11} + \partial_x\hat{\psi}^*\tilde{M}_{N1}\right] \\
&\quad +h^{n+1}\Big[W^n\partial_y\varphi_q + U^n\partial_x\varphi_q\Big]\left[\partial_y\psi_1^n\tilde{M}_{11} + \partial_y\hat{\psi}^*\tilde{M}_{N1}\right] \\
&\quad -\left[\frac{V^n}{W^n}\partial_x h^{n+1}\partial_x\varphi_q + W\partial_y h^{n+1}\partial_y\varphi_q + U^n\left(\partial_x h^{n+1}\partial_y\varphi_q + \partial_y h^{n+1}\partial_x\varphi_q\right)\right] \\
&\qquad \times\left[\tilde{D}_{11}\psi_1^n + \tilde{D}_{1N}\hat{\psi}^{*^T}\right] \\
&\quad -\varphi_q\left[\frac{V^n}{W^n}\partial_x h^{n+1} + U^n\partial_y h^{n+1}\right]\left[\partial_x\psi_1^n\tilde{D}_{11} + \partial_x\hat{\psi}^*\tilde{D}_{N1}\right] \\
&\quad -\varphi_q\Big[W^n\partial_y h^{n+1} + U^n\partial_x h^{n+1}\Big]\left[\partial_y\psi_1^n\tilde{D}_{11} + \partial_y\hat{\psi}^*\tilde{D}_{N1}\right] \\
&\quad +\frac{\varphi_q}{h^{n+1}}\left[\frac{V^n}{W^n}(\partial_x h^{n+1})^2 + W^n(\partial_y h^{n+1})^2 + 2U^n\partial_x h^{n+1}\partial_y h^{n+1}\right]\left[\tilde{S}_{11}\psi_1^n + \hat{\psi}^*\tilde{S}_{N1}\right] \\
&\quad +\frac{W^n H_0^2}{h^{n+1}}\varphi_q\left[\psi_1^n\tilde{A}_{11} + \hat{\psi}^*\tilde{A}_{N1}\right] - H_0(x-L_w)\partial_t\tilde{R}^n\partial_x h^{n+1}\varphi_q \Bigg\}\,\mathrm{d}x\,\mathrm{d}y \\
&+\Delta t\int_0^{L_y}\left(L_w\partial_t\tilde{R}^n h^{n+1}\varphi_q\tilde{I}_1\right)_{x=0}\mathrm{d}y,
\end{aligned} \tag{B.33}$$

simultaneously with

$$
\begin{aligned}
&\int_{\hat{\Omega}_{x,y}} \Bigg\{ h^{n+1} \Bigg[\left(\frac{V^n}{W^n} \partial_x \hat{\varphi} + U^n \partial_y \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_x \hat{\psi}^* + \tilde{M}_{N1} \partial_x \psi_1^n \right) \\
&\qquad\qquad + (W^n \partial_y \hat{\varphi} + U^n \partial_x \hat{\varphi}) \circ \left(\tilde{M}_{NN} \partial_y \hat{\psi}^* + \tilde{M}_{N1} \partial_y \psi_1^n \right) \Bigg] \\
&\quad - \left[\frac{V^n}{W^n} \partial_x h^{n+1} + U^n \partial_y h^{n+1} \right] \Bigg[\partial_x \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{*^T} + \tilde{D}_{N1} \psi_1^n \right) \\
&\qquad\qquad\qquad\qquad + \hat{\varphi}^T \left(\tilde{D}_{NN}^T \partial_x \hat{\psi}^{*^T} + \tilde{D}_{1N}^T \partial_x \psi_1^n \right) \Bigg] \\
&\quad - \left[W^n \partial_y h^{n+1} + U^n \partial_x h^{n+1} \right] \Bigg[\partial_y \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{*^T} + \tilde{D}_{N1} \psi_1^n \right) \\
&\qquad\qquad\qquad\qquad + \hat{\varphi}^T \circ \left(\tilde{D}_{NN}^T \partial_y \hat{\psi}^{*^T} + \tilde{D}_{1N}^T \partial_y \psi_1^n \right) \Bigg] \\
&\quad + \frac{1}{h^{n+1}} \Bigg[\left(\frac{V^n}{W^n} (\partial_x h^{n+1})^2 + W^n (\partial_y h^{n+1})^2 + 2U^n \partial_x h^{n+1} \partial_y h^{n+1} \right) \\
&\qquad\qquad \times \hat{\varphi} \circ \left(\tilde{S}_{NN} \hat{\psi}^{*^T} + \tilde{S}_{N1} \psi_1^n \right) + W^n H_0^2 \hat{\varphi} \circ \left(\tilde{A}_{NN} \hat{\psi}^{*^T} + \tilde{A}_{N1} \psi_1^n \right) \Bigg] \Bigg\} \, \mathrm{d}x \, \mathrm{d}y \\
&+ \int_0^{L_y} \left(L_w h^{n+1} \partial_t \tilde{R}^n \left(\hat{\varphi} \circ \tilde{I}_N \right) \right)_{x=0} \mathrm{d}y = 0.
\end{aligned}
\tag{B.34}
$$

Equation (B.34) is actually a system of n_z equations, each one eliminating the velocity potential in one of the vertical subsurface layers. Finally, the surface velocity potential ψ_1 is updated through

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \varphi_k W^{n+1}\psi_1^{n+1}\,\mathrm{d}x\,\mathrm{d}y = \int_{\hat{\Omega}_{x,y}} & \varphi_k W^n \psi_1^n \\
-\Delta t\Bigg\{ & \frac{\varphi_k}{2}\Bigg[\frac{V^n}{W^n}\Big[(\partial_x\psi_1^n)^2\tilde{M}_{11} + \partial_x\hat{\psi}^*\Big(2\tilde{M}_{N1}\partial_x\psi_1^n + \tilde{M}_{NN}\partial_x\hat{\psi}^{*^T}\Big)\Big] \\
& + W^n\Big[(\partial_y\psi_1^n)^2\tilde{M}_{11} + \partial_y\hat{\psi}^*\Big(2\tilde{M}_{N1}\partial_y\psi_1^n + \tilde{M}_{NN}\partial_y\hat{\psi}^{*^T}\Big)\Big] \\
& + 2U^n\Bigg(\partial_x\psi_1^n\Big(\tilde{M}_{11}\partial_y\psi_1^n + \tilde{M}_{1N}^T\partial_y\hat{\psi}^{*^T}\Big) \\
& \qquad + \partial_x\hat{\psi}^*\Big(\tilde{M}_{N1}\partial_y\psi_1^n + \tilde{M}_{NN}^T\partial_y\hat{\psi}^{*^T}\Big)\Bigg)\Bigg] \\
& - \Bigg[\frac{V^n}{W^n}\partial_x\varphi_k + U^n\partial_y\varphi_k\Bigg] \\
& \times\Bigg[\partial_x\psi_1^n\Big(\tilde{D}_{11}\psi_1^n + \tilde{D}_{1N}\hat{\psi}^{*^T}\Big) + \partial_x\hat{\psi}^*\Big(\tilde{D}_{N1}\psi_1^n + \tilde{D}_{NN}\hat{\psi}^{*^T}\Big)\Bigg] \\
& - \Bigg[W^n\partial_y\varphi_k + U^n\partial_x\varphi_k\Bigg] \\
& \times\Bigg[\partial_y\psi_1^n\Big(\tilde{D}_{11}\psi_1^n + \tilde{D}_{1N}\hat{\psi}^{*^T}\Big) + \partial_y\hat{\psi}^*\Big(\tilde{D}_{N1}\psi_1^n + \tilde{D}_{NN}\hat{\psi}^{*^T}\Big)\Bigg] \\
& + \Bigg[\partial_x\varphi_k\Bigg(\frac{V^n}{W^n}\partial_x h^{n+1} + U^n\partial_y h^{n+1}\Bigg) + \partial_y\varphi_k\left(W^n\partial_y h^{n+1} + U^n\partial_x h^{n+1}\right) \\
& \quad - \frac{\varphi_k}{2h^{n+1}}\Bigg(\frac{V^n}{W^n}(\partial_x h^{n+1})^2 + W^n(\partial_y h^{n+1})^2 + 2U^n\partial_x h^{n+1}\partial_y h^{n+1}\Bigg)\Bigg] \\
& \times\frac{1}{h^{n+1}}\Big[(\psi_1^n)^2\tilde{S}_{11} + \hat{\psi}^*\Big(2\tilde{S}_{N1}\psi_1^n + \tilde{S}_{NN}\hat{\psi}^{*^T}\Big)\Big] \\
& - \frac{H_0^2 W^n}{(h^{n+1})^2}\varphi_k\Big[(\psi_1^n)^2\tilde{A}_{11} + \hat{\psi}^*\Big(2\tilde{A}_{N1}\psi_1^n + \tilde{A}_{NN}\hat{\psi}^{*^T}\Big)\Big] \\
& + H_0\Big[gW^n\varphi_k(h^{n+1} - H) + \psi_1^n(x - L_w)\partial_t\tilde{R}^n\partial_x\varphi_k\Big]\Bigg\}\,\mathrm{d}x\,\mathrm{d}y \\
-\Delta t\int_0^{L_y}\Bigg(\varphi_k L_w\partial_t\tilde{R}^n & \Big(\psi_1^n\tilde{I}_1 + \hat{\psi}^*\tilde{I}_N\Big)\Bigg)_{x=0}\mathrm{d}y.
\end{aligned}
\tag{B.35}
$$

Finally, the interior velocity potential may be updated by solving

$$
\begin{aligned}
&\int_{\hat{\Omega}_{x,y}} \Bigg\{ h^{n+1} \Bigg[\left(\frac{V^{n+1}}{W^{n+1}} \partial_x \hat{\varphi} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_y \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_x \hat{\psi}^{n+1} \right) \\
&\qquad + \left(W^{n+1} \partial_y \hat{\varphi} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_x \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_y \hat{\psi}^{n+1} \right) \Bigg] \\
&\quad - \left[\frac{V^{n+1}}{W^{n+1}} \partial_x h^{n+1} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_y h^{n+1} \right] \\
&\qquad \times \left[\partial_x \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{*^T} \right) + \hat{\varphi}^T \left(\tilde{D}_{NN}^T \partial_x \hat{\psi}^{{n+1}^T} \right) \right] \\
&\quad - \left[W^{n+1} \partial_y h^{n+1} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_x h^{n+1} \right] \\
&\qquad \times \left[\partial_y \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{{n+1}^T} \right) + \hat{\varphi}^T \circ \left(\tilde{D}_{NN}^T \partial_y \hat{\psi}^{{n+1}^T} \right) \right] \\
&\quad + \Bigg[W^{n+1} H_0^2 \hat{\varphi} \circ \left(\tilde{A}_{NN} \hat{\psi}^{{n+1}^T} \right) + \Bigg(\frac{V^{n+1}}{W^{n+1}} (\partial_x h^{n+1})^2 + W^{n+1} (\partial_y h^{n+1})^2 \\
&\qquad + 2(x - L_w) \partial_y \tilde{R}^{n+1} \partial_x h^{n+1} \partial_y h^{n+1} \Bigg) \hat{\varphi} \circ \left(\tilde{S}_{NN} \hat{\psi}^{{n+1}^T} \right) \Bigg] \frac{1}{h^{n+1}} \Bigg\} \, \mathrm{d}x \, \mathrm{d}y \\
= &- \int_{\hat{\Omega}_{x,y}} \Bigg\{ h^{n+1} \Bigg[\left(\frac{V^{n+1}}{W^{n+1}} \partial_x \hat{\varphi} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_y \hat{\varphi} \right) \circ \left(\tilde{M}_{N1} \partial_x \psi_1^{n+1} \right) \\
&\qquad + \left(W^{n+1} \partial_y \hat{\varphi} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_x \hat{\varphi} \right) \circ \left(\tilde{M}_{N1} \partial_y \psi_1^{n+1} \right) \Bigg] \\
&\quad - \left[\frac{V^{n+1}}{W^{n+1}} \partial_x h^{n+1} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_y h^{n+1} \right] \\
&\qquad \times \left[\partial_x \hat{\varphi}^T \circ \left(\tilde{D}_{N1} \psi_1^{n+1} \right) + \hat{\varphi}^T \left(\tilde{D}_{1N}^T \partial_x \psi_1^{n+1} \right) \right] \\
&\quad - \left[W^{n+1} \partial_y h^{n+1} + (x - L_w) \partial_y \tilde{R}^{n+1} \partial_x h^{n+1} \right] \\
&\qquad \times \left[\partial_y \hat{\varphi}^T \circ \left(\tilde{D}_{N1} \psi_1^{n+1} \right) + \hat{\varphi}^T \circ \left(\tilde{D}_{1N}^T \partial_y \psi_1^{n+1} \right) \right] \\
&\quad + \Bigg[W^{n+1} H_0^2 \hat{\varphi} \circ \left(\tilde{A}_{N1} \psi_1^{n+1} \right) + \Bigg(\frac{V^{n+1}}{W^{n+1}} (\partial_x h^{n+1})^2 + W^{n+1} (\partial_y h^{n+1})^2 \\
&\qquad + 2(x - L_w) \partial_y \tilde{R}^{n+1} \partial_x h^{n+1} \partial_y h^{n+1} \Bigg) \hat{\varphi} \circ \left(\tilde{S}_{N1} \psi_1^{n+1} \right) \Bigg] \frac{1}{h^{n+1}} \Bigg\} \, \mathrm{d}x \, \mathrm{d}y \\
&- \int_0^{L_y} \left(L_w h^{n+1} \partial_t \tilde{R}^{n+1} \left(\hat{\varphi} \circ \tilde{I}_N \right) \right)_{x=0} \mathrm{d}y.
\end{aligned}
\tag{B.36}
$$

B.6 Space-continuous-time-discrete weak formulations with the Störmer-Verlet scheme

As for the Symplectic-Euler scheme, Eqns. (B.22) to (B.26) may be written in space-continuous time-discrete vectorial form. The first step is then to solve

$$
\begin{aligned}
&\int_{\hat{\Omega}_{x,y}} \varphi_k\Big(W^{n+1/2}\psi_1^{n+1/2} - W^n\psi_1^n\Big)\\
&\quad+\frac{\Delta t}{2}\Bigg\{\frac{\varphi_k}{2}\Bigg[\frac{V^{n+1/2}}{W^{n+1/2}}\Big[(\partial_x\psi_1^{n+1/2})^2\tilde{M}_{11} + \partial_x\hat{\psi}^*\Big(2\tilde{M}_{N1}\partial_x\psi_1^{n+1/2} + \tilde{M}_{NN}\partial_x\hat{\psi}^{*^T}\Big)\Big]\\
&\qquad\qquad + W^{n+1/2}\Big[(\partial_y\psi_1^{n+1/2})^2\tilde{M}_{11} + \partial_y\hat{\psi}^*\Big(2\tilde{M}_{N1}\partial_y\psi_1^{n+1/2} + \tilde{M}_{NN}\partial_y\hat{\psi}^{*^T}\Big)\Big]\\
&\qquad\qquad + 2U^{n+1/2}\Big(\partial_x\psi_1^{n+1/2}\Big(\tilde{M}_{11}\partial_y\psi_1^{n+1/2} + \tilde{M}_{1N}^T\partial_y\hat{\psi}^{*^T}\Big)\\
&\qquad\qquad\qquad + \partial_x\hat{\psi}^*\Big(\tilde{M}_{N1}\partial_y\psi_1^{n+1/2} + \tilde{M}_{NN}^T\partial_y\hat{\psi}^{*^T}\Big)\Big)\Bigg]\\
&\qquad-\Bigg[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x\varphi_k + U^{n+1/2}\partial_y\varphi_k\Bigg]\Bigg[\partial_x\psi_1^{n+1/2}\Big(\tilde{D}_{11}\psi_1^{n+1/2} + \tilde{D}_{1N}\hat{\psi}^{*^T}\Big)\\
&\qquad\qquad\qquad + \partial_x\hat{\psi}^*\Big(\tilde{D}_{N1}\psi_1^{n+1/2} + \tilde{D}_{NN}\hat{\psi}^{*^T}\Big)\Bigg]\\
&\qquad-\Bigg[W^{n+1/2}\partial_y\varphi_k + U^{n+1/2}\partial_x\varphi_k\Bigg]\Bigg[\partial_y\psi_1^{n+1/2}\Big(\tilde{D}_{11}\psi_1^{n+1/2} + \tilde{D}_{1N}\hat{\psi}^{*^T}\Big)\\
&\qquad\qquad\qquad + \partial_y\hat{\psi}^*\Big(\tilde{D}_{N1}\psi_1^{n+1/2} + \tilde{D}_{NN}\hat{\psi}^{*^T}\Big)\Bigg]\\
&\qquad+\Bigg[\partial_x\varphi_k\Big(\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^n + U^{n+1/2}\partial_y h^n\Big) + \partial_y\varphi_k\Big(W^{n+1/2}\partial_y h^n + U^{n+1/2}\partial_x h^n\Big)\\
&\qquad-\frac{\varphi_k}{2h^n}\Bigg(\frac{V^{n+1/2}}{W^{n+1/2}}(\partial_x h^n)^2 + W^{n+1/2}(\partial_y h^n)^2 + 2U^{n+1/2}\partial_x h^n\partial_y h^n\Bigg)\Bigg]\frac{1}{h^n}\\
&\qquad\times\Big[(\psi_1^{n+1/2})^2\tilde{S}_{11} + \hat{\psi}^*\Big(2\tilde{S}_{N1}\psi_1^{n+1/2} + \tilde{S}_{NN}\hat{\psi}^{*^T}\Big)\Big]\\
&\qquad-\frac{H_0^2W^{n+1/2}}{(h^n)^2}\varphi_k\Big[(\psi_1^{n+1/2})^2\tilde{A}_{11} + \hat{\psi}^*\Big(2\tilde{A}_{N1}\psi_1^{n+1/2} + \tilde{A}_{NN}\hat{\psi}^{*^T}\Big)\Big]\\
&\quad+H_0\Big[g\varphi_k(h^n - H)W^{n+1/2} + \psi_1^{n+1/2}(x - L_w)\partial_t\tilde{R}^{n+1/2}\partial_x\varphi_k\Big]\Bigg\}\,\mathrm{d}x\,\mathrm{d}y\\
&+\frac{\Delta t}{2}\int_0^{L_y}\Big[\varphi_k L_w\partial_t\tilde{R}^{n+1/2}\Big(\psi_1^{n+1/2}\tilde{I}_1 + \hat{\psi}^*\tilde{I}_N\Big)\Big]_{x=0}\mathrm{d}y = 0,
\end{aligned}
$$

to be solved simultaneously with

$$\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \Bigg\{ & h^n \Bigg[\left(\frac{V^{n+1/2}}{W^{n+1/2}} \partial_x \hat{\varphi} + U^{n+1/2} \partial_y \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_x \hat{\psi}^* + \tilde{M}_{N1} \partial_x \psi_1^{n+1/2} \right) \\
& \qquad + \left(W^{n+1/2} \partial_y \hat{\varphi} + U^{n+1/2} \partial_x \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_y \hat{\psi}^* + \tilde{M}_{N1} \partial_y \psi_1^{n+1/2} \right) \Bigg] \\
& - \left[\frac{V^{n+1/2}}{W^{n+1/2}} \partial_x h^n + U^{n+1/2} \partial_y h^n \right] \Bigg[\partial_x \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{*^T} + \tilde{D}_{N1} \psi_1^{n+1/2} \right) \\
& \qquad\qquad + \hat{\varphi}^T \left(\tilde{D}_{NN}^T \partial_x \hat{\psi}^{*^T} + \tilde{D}_{1N}^T \partial_x \psi_1^{n+1/2} \right) \Bigg] \\
& - \left[W^{n+1/2} \partial_y h^n + U^{n+1/2} \partial_x h^n \right] \Bigg[\partial_y \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{*^T} + \tilde{D}_{N1} \psi_1^{n+1/2} \right) \\
& \qquad\qquad + \hat{\varphi}^T \circ \left(\tilde{D}_{NN}^T \partial_y \hat{\psi}^{*^T} + \tilde{D}_{1N}^T \partial_y \psi_1^{n+1/2} \right) \Bigg] \\
& + \frac{1}{h^n} \Bigg[W^{n+1/2} H_0^2 \hat{\varphi} \circ \left(\tilde{A}_{NN} \hat{\psi}^{*^T} + \tilde{A}_{N1} \psi_1^{n+1/2} \right) + \hat{\varphi} \Bigg(\frac{V^{n+1/2}}{W^{n+1/2}} (\partial_x h^n)^2 \\
& \qquad + W^{n+1/2} (\partial_y h^n)^2 + 2U^{n+1/2} \partial_x h^n \partial_y h^n \Bigg) \circ \left(\tilde{S}_{NN} \hat{\psi}^{*^T} + \tilde{S}_{N1} \psi_1^{n+1/2} \right) \Bigg] \Bigg\} \, \mathrm{d}x \, \mathrm{d}y \\
& + \int_0^{L_y} \left(L_w h^n \partial_t \tilde{R}^{n+1/2} \left(\hat{\varphi} \circ \tilde{I}_N \right) \right)_{x=0} \mathrm{d}y = 0.
\end{aligned} \tag{B.38}$$

The second step, corresponding to Eqn. B.24 is implemented as

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} & W^{n+1/2}\varphi_q h^{n+1}\,\mathrm{d}x\,\mathrm{d}y = \int_{\hat{\Omega}_{x,y}} W^{n+1/2}\varphi_q h^n \\
+ \frac{\Delta t}{2}\Bigg\{ & \left[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x\varphi_q + U^{n+1/2}\partial_y\varphi_q\right]\left[h^n\left(\partial_x\psi_1^{n+1/2}\tilde{M}_{11} + \partial_x\hat{\psi}^*\tilde{M}_{N1}\right)\right. \\
& \qquad \left. + h^{n+1}\left(\partial_x\psi_1^{n+1/2}\tilde{M}_{11} + \partial_x\hat{\psi}^{**}\tilde{M}_{N1}\right)\right] \\
& + \left[W^{n+1/2}\partial_y\varphi_q + U^{n+1/2}\partial_x\varphi_q\right]\left[h^n\left(\partial_y\psi_1^{n+1/2}\tilde{M}_{11} + \partial_y\hat{\psi}^*\tilde{M}_{N1}\right)\right. \\
& \qquad \left. + h^{n+1}\left(\partial_y\psi_1^{n+1/2}\tilde{M}_{11} + \partial_y\hat{\psi}^{**}\tilde{M}_{N1}\right)\right] \\
& - \left[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^{n+1}\partial_x\varphi_q + W\partial_y h^{n+1}\partial_y\varphi_q\right. \\
& \quad \left. + U^{n+1/2}\left(\partial_x h^{n+1}\partial_y\varphi_q + \partial_y h^{n+1}\partial_x\varphi_q\right)\right]\left[\tilde{D}_{11}\psi_1^{n+1/2} + \tilde{D}_{1N}\hat{\psi}^{**^T}\right] \\
& - \left[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^n\partial_x\varphi_q + W\partial_y h^n\partial_y\varphi_q\right. \\
& \quad \left. + U^{n+1/2}\left(\partial_x h^n\partial_y\varphi_q + \partial_y h^n\partial_x\varphi_q\right)\right]\left[\tilde{D}_{11}\psi_1^{n+1/2} + \tilde{D}_{1N}\hat{\psi}^{*^T}\right] \\
& - \varphi_q\left[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^{n+1} + U^{n+1/2}\partial_y h^{n+1}\right]\left[\partial_x\psi_1^{n+1/2}\tilde{D}_{11} + \partial_x\hat{\psi}^{**}\tilde{D}_{N1}\right] \\
& - \varphi_q\left[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^n + U^{n+1/2}\partial_y h^n\right]\left[\partial_x\psi_1^{n+1/2}\tilde{D}_{11} + \partial_x\hat{\psi}^*\tilde{D}_{N1}\right] \\
& - \varphi_q\left[W^{n+1/2}\partial_y h^{n+1} + U^{n+1/2}\partial_x h^{n+1}\right]\left[\partial_y\psi_1^{n+1/2}\tilde{D}_{11} + \partial_y\hat{\psi}^{**}\tilde{D}_{N1}\right] \\
& - \varphi_q\left[W^{n+1/2}\partial_y h^n + U^{n+1/2}\partial_x h^n\right]\left[\partial_y\psi_1^{n+1/2}\tilde{D}_{11} + \partial_y\hat{\psi}^*\tilde{D}_{N1}\right] \\
& + \frac{\varphi_q}{h^{n+1}}\left[\frac{V^{n+1/2}}{W^{n+1/2}}(\partial_x h^{n+1})^2 + W^{n+1/2}(\partial_y h^{n+1})^2\right. \\
& \qquad \left. + 2U^{n+1/2}\partial_x h^{n+1}\partial_y h^{n+1}\right]\left[\tilde{S}_{11}\psi_1^{n+1/2} + \hat{\psi}^{**}\tilde{S}_{N1}\right] \\
& + \frac{\varphi_q}{h^n}\left[\frac{V^{n+1/2}}{W^{n+1/2}}(\partial_x h^n)^2 + W^{n+1/2}(\partial_y h^n)^2 + 2U^{n+1/2}\partial_x h^n\partial_y h^n\right] \\
& \quad \times\left[\tilde{S}_{11}\psi_1^{n+1/2} + \hat{\psi}^*\tilde{S}_{N1}\right] + \varphi_q\left[\frac{1}{h^{n+1}}\left(\psi_1^{n+1/2}\tilde{A}_{11} + \hat{\psi}^{**}\tilde{A}_{N1}\right)\right. \\
& \qquad \left. + \frac{1}{h^n}\left(\psi_1^{n+1/2}\tilde{A}_{11} + \hat{\psi}^*\tilde{A}_{N1}\right)\right]W^{n+1/2}H_0^2 \\
& - H_0(x - L_w)\partial_t\tilde{R}^{n+1/2}\varphi_q\left(\partial_x h^{n+1} + \partial_x h^n\right)\Bigg\}\,\mathrm{d}x\,\mathrm{d}y \\
+ \frac{\Delta t}{2} & \int_0^{L_y}\left(L_w\partial_t\tilde{R}^{n+1/2}\varphi_q\tilde{I}_1\left(h^{n+1} + h^n\right)\right)_{x=0}\mathrm{d}y,
\end{aligned}
\tag{B.39}
$$

to be solved simultaneously with

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \Bigg\{ & h^{n+1}\Bigg[\left(\frac{V^{n+1/2}}{W^{n+1/2}} \partial_x \hat{\varphi} + U^{n+1/2} \partial_y \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_x \hat{\psi}^{**} + \tilde{M}_{N1} \partial_x \psi_1^{n+1/2} \right) \\
& \qquad + \left(W^{n+1/2} \partial_y \hat{\varphi} + U^{n+1/2} \partial_x \hat{\varphi} \right) \circ \left(\tilde{M}_{NN} \partial_y \hat{\psi}^{**} + \tilde{M}_{N1} \partial_y \psi_1^{n+1/2} \right) \Bigg] \\
& - \left[\frac{V^{n+1/2}}{W^{n+1/2}} \partial_x h^{n+1} + U^{n+1/2} \partial_y h^{n+1} \right] \Bigg[\partial_x \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{**^T} + \tilde{D}_{N1} \psi_1^{n+1/2} \right) \\
& \qquad + \hat{\varphi}^T \left(\tilde{D}_{NN}^T \partial_x \hat{\psi}^{**^T} + \tilde{D}_{1N}^T \partial_x \psi_1^{n+1/2} \right) \Bigg] \\
& - \left[W^{n+1/2} \partial_y h^{n+1} + U^{n+1/2} \partial_x h^{n+1} \right] \Bigg[\partial_y \hat{\varphi}^T \circ \left(\tilde{D}_{NN} \hat{\psi}^{**^T} + \tilde{D}_{N1} \psi_1^{n+1/2} \right) \\
& \qquad + \hat{\varphi}^T \circ \left(\tilde{D}_{NN}^T \partial_y \hat{\psi}^{**^T} + \tilde{D}_{1N}^T \partial_y \psi_1^{n+1/2} \right) \Bigg] \\
& + \frac{1}{h^{n+1}} \Bigg[W^{n+1/2} H_0^2 \hat{\varphi} \circ \left(\tilde{A}_{NN} \hat{\psi}^{**^T} + \tilde{A}_{N1} \psi_1^{n+1/2} \right) \\
& \qquad + \Bigg(\frac{V^{n+1/2}}{W^{n+1/2}} (\partial_x h^{n+1})^2 + W^{n+1/2} (\partial_y h^{n+1})^2 \\
& \qquad\quad + 2U^{n+1/2} \partial_x h^{n+1} \partial_y h^{n+1} \Bigg) \hat{\varphi} \circ \left(\tilde{S}_{NN} \hat{\psi}^{**^T} + \tilde{S}_{N1} \psi_1^{n+1/2} \right) \Bigg] \Bigg\} \, \mathrm{d}x \, \mathrm{d}y \\
+ \int_0^{L_y} & \left(L_w h^{n+1} \partial_t \tilde{R}^{n+1/2} \left(\hat{\varphi} \circ \tilde{I}_N \right) \right)_{x=0} \mathrm{d}y = 0.
\end{aligned}
\tag{B.40}
$$

Finally, the last step is

$$
\begin{aligned}
&\int_{\hat{\Omega}_{x,y}} \varphi_k W^{n+1}\psi_1^{n+1}\,\mathrm{d}x\,\mathrm{d}y = \int_{\hat{\Omega}_{x,y}} \varphi_k W^{n+1/2}\psi_1^{n+1/2} \\
&-\frac{\Delta t}{2}\Bigg\{\frac{\varphi_k}{2}\Bigg[\frac{V^{n+1/2}}{W^{n+1/2}}\Big[(\partial_x\psi_1^{n+1/2})^2\tilde{M}_{11}+\partial_x\hat{\psi}^{**}\Big(2\tilde{M}_{N1}\partial_x\psi_1^{n+1/2}+\tilde{M}_{NN}\partial_x\hat{\psi}^{**^T}\Big)\Big] \\
&\qquad +W^{n+1/2}\Big[(\partial_y\psi_1^{n+1/2})^2\tilde{M}_{11}+\partial_y\hat{\psi}^{**}\Big(2\tilde{M}_{N1}\partial_y\psi_1^{n+1/2}+\tilde{M}_{NN}\partial_y\hat{\psi}^{**^T}\Big)\Big] \\
&\qquad +2U^{n+1/2}\Big[\partial_x\psi_1^{n+1/2}\Big(\tilde{M}_{11}\partial_y\psi_1^{n+1/2}+\tilde{M}_{1N}^T\partial_y\hat{\psi}^{**^T}\Big) \\
&\qquad\qquad +\partial_x\hat{\psi}^{**}\Big(\tilde{M}_{N1}\partial_y\psi_1^{n+1/2}+\tilde{M}_{NN}^T\partial_y\hat{\psi}^{**^T}\Big)\Big]\Bigg] \\
&\quad -\Big[\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x\varphi_k+U^{n+1/2}\partial_y\varphi_k\Big]\Big[\partial_x\psi_1^{n+1/2}\Big(\tilde{D}_{11}\psi_1^{n+1/2}+\tilde{D}_{1N}\hat{\psi}^{**^T}\Big) \\
&\qquad\qquad +\partial_x\hat{\psi}^{**}\Big(\tilde{D}_{N1}\psi_1^{n+1/2}+\tilde{D}_{NN}\hat{\psi}^{**^T}\Big)\Big] \\
&\quad -\Big[W^{n+1/2}\partial_y\varphi_k+U^{n+1/2}\partial_x\varphi_k\Big]\Big[\partial_y\psi_1^{n+1/2}\Big(\tilde{D}_{11}\psi_1^{n+1/2}+\tilde{D}_{1N}\hat{\psi}^{**^T}\Big) \\
&\qquad\qquad +\partial_y\hat{\psi}^{**}\Big(\tilde{D}_{N1}\psi_1^{n+1/2}+\tilde{D}_{NN}\hat{\psi}^{**^T}\Big)\Big] \\
&\quad +\frac{1}{h^{n+1}}\Big[\partial_x\varphi_k\Big(\frac{V^{n+1/2}}{W^{n+1/2}}\partial_x h^{n+1}+U^{n+1/2}\partial_y h^{n+1}\Big) \\
&\qquad +\partial_y\varphi_k\Big(W^{n+1/2}\partial_y h^{n+1}+U^{n+1/2}\partial_x h^{n+1}\Big) \\
&\qquad -\frac{\varphi_k}{2h^{n+1}}\Big(\frac{V^{n+1/2}}{W^{n+1/2}}(\partial_x h^{n+1})^2+W^{n+1/2}(\partial_y h^{n+1})^2 \\
&\qquad\qquad +2U^{n+1/2}\partial_x h^{n+1}\partial_y h^{n+1}\Big)\Big] \\
&\qquad \times\Big[(\psi_1^{n+1/2})^2\tilde{S}_{11}+\hat{\psi}^{**}\Big(2\tilde{S}_{N1}\psi_1^{n+1/2}+\tilde{S}_{NN}\hat{\psi}^{**^T}\Big)\Big] \\
&\quad -\frac{H_0^2W^{n+1/2}}{(h^{n+1})^2}\varphi_k\Big[(\psi_1^{n+1/2})^2\tilde{A}_{11}+\hat{\psi}^{**}\Big(2\tilde{A}_{N1}\psi_1^{n+1/2}+\tilde{A}_{NN}\hat{\psi}^{**^T}\Big)\Big] \\
&\quad +H_0\Big[gW^{n+1/2}\varphi_k(h^{n+1}-H)+\psi_1^{n+1/2}(x-L_w)\partial_t\tilde{R}^{n+1/2}\partial_x\varphi_k\Big]\Bigg\}\,\mathrm{d}x\,\mathrm{d}y \\
&-\frac{\Delta t}{2}\int_0^{L_y}\Big(\varphi_k L_w\partial_t\tilde{R}^{n+1/2}\Big(\psi_1^{n+1/2}\tilde{I}_1+\hat{\psi}^{**}\tilde{I}_N\Big)\Big)_{x=0}\mathrm{d}y.
\end{aligned}
\tag{B.41}
$$

Again, the interior velocity potential may be updated with Eqn. (B.36).

B.7 Weak formulations obtained from the continuous variational principle

In this section we show that the weak formulations (B.33) to (B.35) and (B.37) to (B.41) may be obtained from the variations of h, ψ_1 and $\psi_{i'}$ in the variational principle (3.27). Taking the variations of h, ψ_1 and $\psi_{i'}$ in the variational principle (3.27) and setting respectively ($\delta h \neq 0$, $\delta\psi_1 = 0$, $\delta\hat{\psi} = 0$), ($\delta h = 0$, $\delta\psi_1 \neq 0$, $\delta\hat{\psi} = 0$) and ($\delta h = 0$, $\delta\psi_1 = 0$, $\delta\hat{\psi} \neq 0$), indeed leads to the following weak formulations:

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \Bigg\{ \delta h \Bigg[& \frac{V}{2W} \Big[(\partial_x\psi_1)^2 \tilde{M}_{11} + \partial_x\hat{\psi} \Big(2\tilde{M}_{N1}\partial_x\psi_1 + \tilde{M}_{NN}(\partial_x\hat{\psi})^T \Big) \Big] \\
& + \frac{W}{2} \Big[(\partial_y\psi_1)^2 \tilde{M}_{11} + \partial_y\hat{\psi} \Big(2\tilde{M}_{N1}\partial_y\psi_1 + \tilde{M}_{NN}(\partial_y\hat{\psi})^T \Big) \Big] \\
& + U \Big[\partial_x\psi_1 \Big(\tilde{M}_{11}\partial_y\psi_1 + \tilde{M}_{1N}\partial_y\hat{\psi}^T \Big) + \partial_x\hat{\psi} \Big(\tilde{M}_{N1}\partial_y\psi_1 + \tilde{M}_{NN}\partial_y\hat{\psi}^T \Big) \Big] \Bigg] \\
& - \Big[\frac{V}{W}\partial_x\delta h + U\partial_y\delta h \Big] \Big[\partial_x\psi_1 \Big(\tilde{D}_{11}\psi_1 + \tilde{D}_{1N}\hat{\psi}^T \Big) + \partial_x\hat{\psi} \Big(\tilde{D}_{N1}\psi_1 + \tilde{D}_{NN}\hat{\psi}^T \Big) \Big] \\
& - \left[W\partial_y\delta h + U\partial_x\delta h \right] \Big[\partial_y\psi_1 \Big(\tilde{D}_{11}\psi_1 + \tilde{D}_{1N}\hat{\psi}^T \Big) + \partial_y\hat{\psi} \Big(\tilde{D}_{N1}\psi_1 + \tilde{D}_{NN}\hat{\psi}^T \Big) \Big] \\
& + \frac{1}{h} \Bigg[\partial_x\delta h \Big(\frac{V}{W}\partial_x h + \partial_y h U \Big) - \frac{\delta h}{h} \Big(\frac{V}{2W}(\partial_x h)^2 + \frac{W}{2}(\partial_y h)^2 + U\partial_x h\partial_y h \Big) \\
& \qquad + \partial_y\delta h \left(W\partial_y h + U\partial_x h \right) \Bigg] \Big[\psi_1^2\tilde{S}_{11} + 2\hat{\psi}\tilde{S}_{N1}\psi_1 + \hat{\psi}\tilde{S}_{NN}\hat{\psi}^T \Big] \\
& - \delta h \frac{WH_0^2}{2h^2} \Big(\psi_1^2\tilde{A}_{11} + 2\hat{\psi}\tilde{A}_{N1}\psi_1 + \hat{\psi}\tilde{A}_{NN}\hat{\psi}^T \Big) \\
& + H_0 \Big(gW\delta h(h-H) - \psi_1(x-L_w)\partial_t\tilde{R}\partial_x\delta h + \partial_t(\psi_1 W)\delta h \Big)_{z=H_0} \Bigg\} \mathrm{d}x\,\mathrm{d}y \\
+ \int_0^{L_y} & \Big(L_w\partial_t\tilde{R}\delta h \Big[\psi_1\tilde{I}_1 + \hat{\psi}\tilde{I}_N \Big] \Big)_{x=0} \mathrm{d}y = 0,
\end{aligned}
\tag{B.42}
$$

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \Bigg\{ & \frac{hV}{W}\left[\partial_x\psi_1\tilde{M}_{11}+\partial_x\hat{\psi}\tilde{M}_{N1}\right]\partial_x\delta\psi_1 + Wh\left[\partial_y\psi_1\tilde{M}_{11}+\partial_y\hat{\psi}\tilde{M}_{N1}\right]\partial_y\delta\psi_1 \\
& + Uh\Bigg[\partial_x\delta\psi_1\left(\tilde{M}_{11}\partial_y\psi_1+\tilde{M}_{1N}\partial_y\hat{\psi}^T\right)+\left(\partial_x\psi_1\tilde{M}_{11}+\partial_x\hat{\psi}\tilde{M}_{N1}\right)\partial_y\delta\psi_1\Bigg] \\
& -\Big[\frac{V}{W}\partial_x h+U\partial_y h\Big]\left[\partial_x\delta\psi_1\left(\tilde{D}_{11}\psi_1+\tilde{D}_{1N}\hat{\psi}^T\right)+\left(\partial_x\psi_1\tilde{D}_{11}+\partial_x\hat{\psi}\tilde{D}_{N1}\right)\delta\psi_1\right] \\
& -\left[W\partial_y h+U\partial_x h\right]\left[\partial_y\delta\psi_1\left(\tilde{D}_{11}\psi_1+\tilde{D}_{1N}\hat{\psi}^T\right)+\left(\partial_y\psi_1\tilde{D}_{11}+\partial_y\hat{\psi}\tilde{D}_{N1}\right)\delta\psi_1\right] \\
& +\frac{WH_0^2}{h}\left(\psi_1\tilde{A}_{11}+\hat{\psi}\tilde{A}_{N1}\right)\delta\psi_1-\delta\psi_1 H_0\left((x-L_w)\partial_t R\partial_x h+W\partial_t h\right)_{z=H_0} \\
& +\frac{1}{h}\Big(\frac{V}{W}(\partial_x h)^2+W(\partial_y h)^2+2U\partial_x h\partial_y h\Big)\left[\psi_1\tilde{S}_{11}+\hat{\psi}\tilde{S}_{N1}\right]\delta\psi_1\Bigg\}\,\mathrm{d}x\,\mathrm{d}y \\
+\int_0^{L_y} & \left(\delta\psi_1 L_w\partial_t\tilde{R}h\tilde{I}_1\right)_{x=0}\mathrm{d}y\Bigg\}\,\mathrm{d}t=0,
\end{aligned}
\tag{B.43}
$$

and

$$
\begin{aligned}
\int_{\hat{\Omega}_{x,y}} \Bigg\{ & \frac{hV}{W}\left[\partial_x\delta\hat{\psi}\circ\left(\tilde{M}_{N1}\partial_x\psi_1+\tilde{M}_{NN}(\partial_x\hat{\psi})^T\right)\right] \\
& + Wh\left[\partial_y\delta\hat{\psi}\circ\left(\tilde{M}_{N1}\partial_y\psi_1+\tilde{M}_{NN}(\partial_y\hat{\psi})^T\right)\right] \\
& + Uh\Bigg[\partial_y\delta\hat{\psi}^T\circ\left(\tilde{M}_{N1}\partial_x\psi_1+\tilde{M}_{NN}\partial_x\hat{\psi}\right)+\partial_x\delta\hat{\psi}^T\left(\tilde{M}_{N1}\partial_y\psi_1+\tilde{M}_{NN}\partial_y\hat{\psi}\right)\Bigg] \\
& -\left[\frac{V}{W}\partial_x h+U\partial_y h\right]\Bigg[\delta\hat{\psi}^T\circ\left(\tilde{D}_{1N}^T\partial_x\psi_1+\tilde{D}_{NN}^T\partial_x\hat{\psi}^T\right) \\
& \qquad\qquad +\partial_x\delta\hat{\psi}\circ\left(\tilde{D}_{N1}\psi_1+\tilde{D}_{NN}\hat{\psi}^T\right)\Bigg] \\
& -\left(W\partial_y h+U\partial_x h\right)\Bigg[\delta\hat{\psi}^T\circ\left(\tilde{D}_{1N}^T\partial_y\psi_1+\tilde{D}_{NN}^T\partial_y\hat{\psi}^T\right) \\
& \qquad\qquad +\partial_y\delta\hat{\psi}\circ\left(\tilde{D}_{N1}\psi_1+\tilde{D}_{NN}\hat{\psi}^T\right)\Bigg] \\
& +\frac{1}{h}\Big(\frac{V}{W}(\partial_x h)^2+W(\partial_y h)^2+2U\partial_x h\partial_y h\Big)\delta\hat{\psi}\circ\left[\tilde{S}_{N1}\psi_1+\tilde{S}_{NN}\hat{\psi}^T\right] \\
& +\frac{WH_0^2}{h}\delta\hat{\psi}\circ\left(\tilde{A}_{N1}\psi_1+\tilde{A}_{NN}\hat{\psi}^T\right)\Bigg)\,\mathrm{d}x\,\mathrm{d}y \\
+\int_0^{L_y} & \left(L_w\partial_t\tilde{R}h\delta\hat{\psi}\circ\tilde{I}_N\right)_{x=0}\Bigg\}\,\mathrm{d}y\,\mathrm{d}t=0.
\end{aligned}
\tag{B.44}
$$

The temporal schemes obtained in sections 3.4.2 and 3.4.3 may be applied directly to the weak formulations (B.42), (B.43) and (B.44).

B.8 Wavemaker motion and velocity as used in section 3.7.1

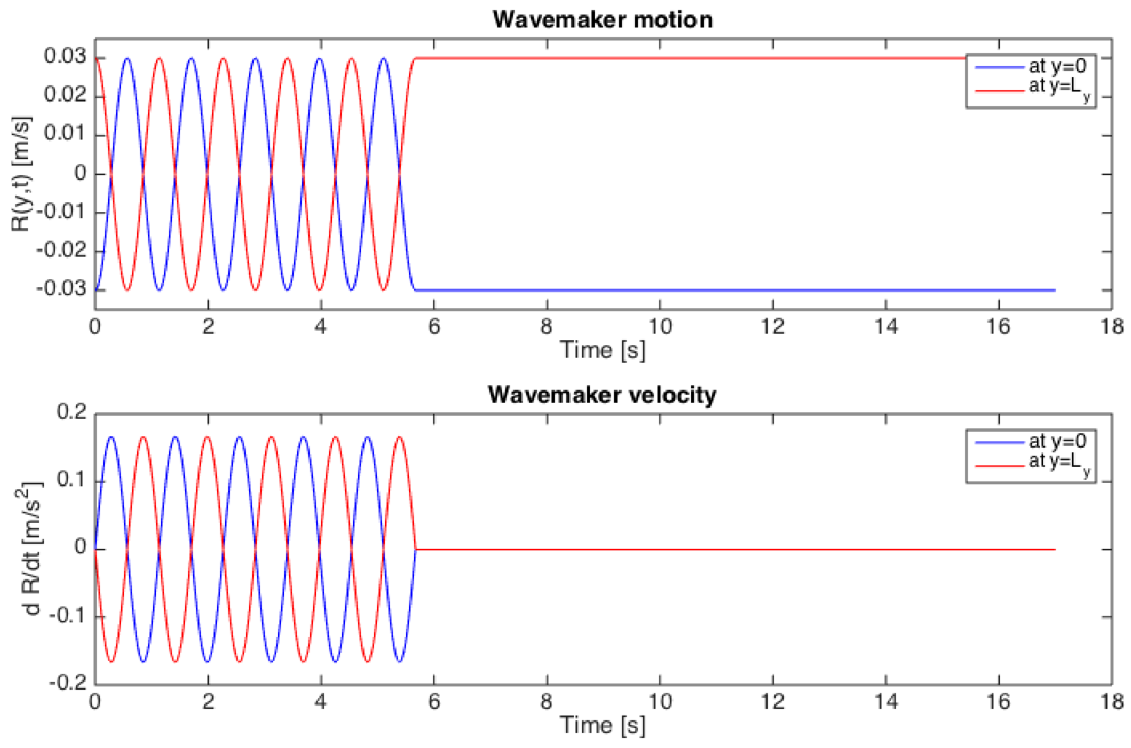


Figure B.4: Evolution of the wavemaker motion (top) and velocity (bottom) at $y = 0$ (blue) and $y = L_y$ (red) in the test of energy conservation.

B.9 Fast Fourier transform of the experimental free-surface elevation

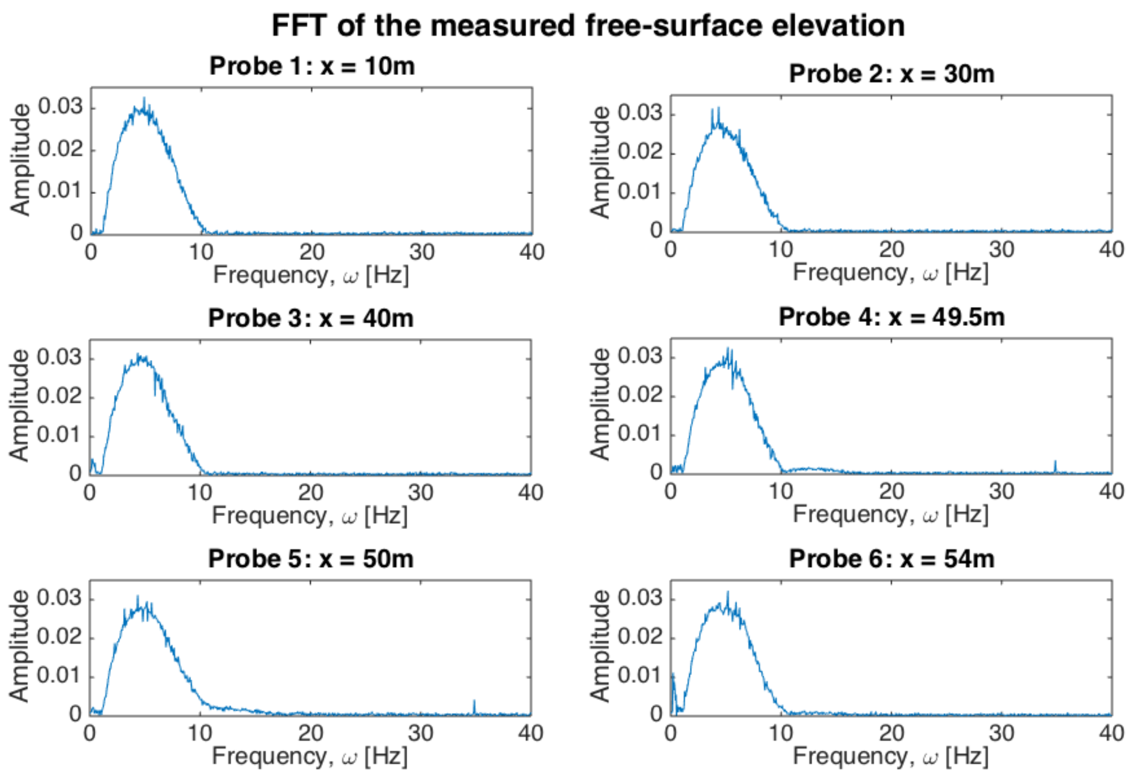


Figure B.5: Fast Fourier transform of the measured free-surface elevation at the probes $x_1 = 10$m, $x_2 = 20$m, $x_3 = 40$m, $x_4 = 49.5$m, $x_5 = 50$m and $x_6 = 54$m.

C Numerical wave tank for offshore applications: dynamics of wavemaker, wave propagation and absorbing waves

C.1 Harten-Lax-van Leer (HLL) flux for the Godunov scheme

The HLL numerical flux is computed from the left and right wave speeds S_L and S_R, which are obtained from the eigenvalues of the system of equations. Setting the left and right values of the flux $F_{k+1/2}$ at the interface $x_{k+1/2}$ as [86]

$$\mathbf{F}_L = \mathbf{F}(\mathbf{U}_{k+1/2^-}(t)) \quad \text{and} \quad \mathbf{F}_R = \mathbf{F}(\mathbf{U}_{k+1/2^+}(t)), \tag{C.45}$$

the numerical flux $\mathbf{F}_{k+1/2}$ is computed as

$$\mathbf{F}_{k+1/2} = \begin{cases} \mathbf{F}_L & \text{if } 0 < S_L, \\ \dfrac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_LS_R(\mathbf{U}_R 0 \mathbf{U}_L)}{S_R - S_L} & \text{if } S_L \le 0 \le S_R, \\ \mathbf{F}_R & \text{if } S_R < 0, \end{cases} \tag{C.46}$$

where S_L and S_R are the left and right wave speeds given by

$$S_L(\boldsymbol{U}_{k+1/2}) = \min\left(\frac{hu^n_{k+1/2^-}}{h^n_{k+1/2^-}} - \sqrt{gh^n_{k+1/2^-}}, \frac{hu^n_{k+1/2^+}}{h^n_{k+1/2^+}} + \sqrt{gh^n_{k+1/2^+}}\right), \tag{C.47}$$

$$S_R(\boldsymbol{U}_{k+1/2}) = \max\left(\frac{hu^n_{k+1/2^-}}{h^n_{k+1/2^-}} + \sqrt{gh^n_{k+1/2^-}}, \frac{hu^n_{k+1/2^+}}{h^n_{k+1/2^+}} + \sqrt{gh^n_{k+1/2^+}}\right). \tag{C.48}$$

C.2 Space-discrete matrices

In the space-discrete variational principle (4.39), the spatial matrices are defined by

$$\begin{aligned} A_{kmq} &= \int_0^{x_c} \left[\frac{L_w}{W}\varphi_k\partial_x\varphi_m\partial_x\varphi_q\right] \mathrm{d}x, \\ M_{kl} &= \int_0^{x_c} \left[\frac{W}{L_w}\varphi_k\varphi_l\right] \mathrm{d}x, \\ I_k &= \int_0^{x_c} \left[\frac{W}{L_w}\varphi_k\right] \mathrm{d}x, \\ E_{kmqlp} &= \int_0^{x_c} \left[\frac{L_w}{W}\frac{1}{\varphi_k}\varphi_m\varphi_q\partial_x\varphi_l\partial_x\varphi_p\right] \mathrm{d}x, \end{aligned}$$

$$\begin{aligned} N_{qk} &= \int_0^{x_c} \left[(x - L_w)\partial_t\tilde{R}\varphi_q\partial_x\varphi_k\right] \mathrm{d}x, \\ J_{kmq} &= \int_0^{x_c} \left[\frac{W}{L_w\varphi_k}\varphi_m\varphi_q\right] \mathrm{d}x, \\ X_{kq} &= \left[\partial_t\tilde{R}\varphi_k\varphi_q\right]_{x=0}. \end{aligned}$$

C.3 Transformed deep-water weak formulations and equations

The deep-water weak formulations with coupling term are given in (4.50a-b). These weak formulations describe the temporal evolution of h and ψ_i. In particular, (4.50a) is a system of n_z+1 equations, that is, one for $i=1$ which describes the evolution of h through

$$\begin{aligned}\int_0^{x_c}\varphi_q\frac{W}{L_w}\partial_t h\mathrm{d}x = \int_0^{x_c}\Bigg\{&\frac{1}{\Upsilon}\Bigg[h\partial_x\varphi_q\partial_x\psi_j\tilde{M}_{1j}+\frac{1}{h}(\partial_x h)^2\psi_j\varphi_q\tilde{S}_{1j}\\ &-\partial_x h\Big(\partial_x\varphi_q\tilde{D}_{1j}\psi_j+\varphi_q\partial_x\psi_j\tilde{D}_{j1}\Big)\Bigg]\\ &+\Upsilon\left[\frac{1}{h}\varphi_q\tilde{A}_{ij}\psi_j-\frac{1}{H_0}\varphi_q\frac{(x-L_w)}{W}\partial_t\tilde{R}\partial_x h\right]\Bigg\}\mathrm{d}x\\ +\frac{1}{H_0}\tilde{I}_1\Bigg\{&\Big(h\partial_t R\Big)_{x=0}-\Big(hu\Big)_{x=x_c}\Bigg\},\end{aligned} \tag{C.49}$$

and n_z for $i'\in[2,n_z+1]$ that express the interior velocity potential $\psi_{i'}$ in terms of the surface velocity potential ψ_1 and the depth h through:

$$\begin{aligned}&\int_0^{x_c}\Bigg\{\frac{1}{\Upsilon}\Bigg[h\partial_x\varphi_q\partial_x\psi_{j'}\tilde{M}_{i'j'}+\frac{1}{h}(\partial_x h)^2\psi_{j'}\varphi_q\tilde{S}_{i'j'}\\ &\qquad-\partial_x h\Big(\partial_x\varphi_q\tilde{D}_{i'j'}\psi_{j'}+\varphi_q\partial_x\psi_{j'}\tilde{D}_{j'i'}\Big)\Bigg]+\Upsilon\frac{1}{h}\varphi_q\tilde{A}_{i'j'}\psi_{j'}\Bigg\}\mathrm{d}x\\ =&-\int_0^{x_c}\Bigg\{\frac{1}{\Upsilon}\Bigg[h\partial_x\varphi_q\partial_x\psi_1\tilde{M}_{i'1}-\partial_x h\Big(\partial_x\varphi_q\tilde{D}_{i'1}\psi_1+\varphi_q\partial_x\psi_1\tilde{D}_{1i'}\Big)\\ &\qquad+\frac{1}{h}(\partial_x h)^2\psi_1\varphi_q\tilde{S}_{i'1}\Bigg]+\Upsilon\frac{1}{h}\varphi_q\tilde{A}_{i'1}\psi_1\Bigg\}\mathrm{d}x\\ &-\frac{1}{H_0}\tilde{I}_{i'}\Bigg\{(h\partial_t R)_{x=0}-(hu)_{x=x_c}\Bigg\},\end{aligned} \tag{C.50}$$

Integrations by part in space of the deep-water weak formulations lead to the transformed nonlinear potential-flow equations with coupling boundary condition at $x = x_c$ as:

$$\begin{aligned}\frac{W}{L_w}\partial_t h =&\frac{1}{\Upsilon}\left[\frac{1}{h}(\partial_x h)^2\psi_j\tilde{S}_{ij} - \partial_x h\partial_x\psi_j\tilde{D}_{ji}\right] - \partial_x\left[\frac{1}{\Upsilon}\left(h\partial_x\psi_j\tilde{M}_{ij} - \partial_x h\tilde{D}_{ij}\psi_j\right)\right]\\ &+\Upsilon\left[\frac{1}{h}\tilde{A}_{ij}\psi_j - \frac{\delta_{i1}}{H_0}\frac{x-L_w}{W}\partial_t\tilde{R}\partial_x h\right], \qquad \forall x\in[0,x_c],\end{aligned} \tag{C.51a}$$

$$\begin{aligned}\frac{1}{L_w}\partial_t(W\psi_1) =& -\frac{1}{2\Upsilon}\left[\partial_x\psi_i\tilde{M}_{ij}\partial_x\psi_j - \frac{1}{h^2}(\partial_x h)^2\psi_i\tilde{S}_{ij}\psi_j\right]\\ &+\Upsilon\left[-\frac{1}{H_0}g(h-H) + \frac{1}{2h^2}\psi_i\tilde{A}_{ij}\psi_j\right]\\ &+\partial_x\left[\frac{1}{\Upsilon}\left(-\partial_x\psi_i\tilde{D}_{ij}\psi_j + \frac{1}{h}\partial_x h\psi_i\tilde{S}_{ij}\psi_j\right) - \frac{\Upsilon}{H_0}\left(\psi_1\frac{(x-L_w)}{W}\partial_t\tilde{R}\right)\right]\end{aligned} \tag{C.51b}$$

$$\frac{1}{H_0}\left[h\partial_x\psi_j\tilde{M}_{ij} - \partial_x h\tilde{D}_{ij}\psi_j\right] = \frac{1}{H_0}\tilde{I}_i hu, \qquad \forall i\in[1,n_z+1],\ \text{at } x=x_c, \tag{C.51c}$$

$$\frac{1}{H_0}\left[\partial_x\psi_i\tilde{D}_{ij}\psi_j - \frac{1}{h}\partial_x h\psi_i\tilde{S}_{ij}\psi_j\right] = \frac{1}{H_0 h}\tilde{G}_i\psi_i hu, \qquad \text{at } x=x_c, \tag{C.51d}$$

$$\frac{1}{H_0}\left[h\partial_x\psi_j\tilde{M}_{ij} - \partial_x h\tilde{D}_{ij}\psi_j\right] = \frac{1}{H_0}h\partial_t R, \qquad \text{at } x=0. \tag{C.51e}$$

Note that Eqns. (C.51c) and (C.51d) are equivalent and correspond to the transform of the coupling condition (4.15) for each unknown.

C.4 Deep- and shallow-water Hamiltonians

The space-discrete variational principle (4.39) may be written in Hamiltonian form (4.57) with the deep-water Hamiltonian

$$\begin{aligned}H(\boldsymbol{h},\boldsymbol{p}_i,W) =&\frac{1}{2H_0}\Big[h_l A_{lqm}M^{-1}_{q\alpha}p_{i\alpha}M^{-1}_{m\beta}p_{j\beta}\tilde{M}_{ij} + \frac{h_l h_n}{h_p}M^{-1}_{q\alpha}p_{i\alpha}\tilde{S}_{ij}M^{-1}_{m\beta}p_{j\beta}E_{pmqln}\\ &- 2A_{mlq}h_l M^{-1}_{q\alpha}p_{i\alpha}\tilde{D}_{ij}M^{-1}_{m\beta}p_{j\beta}\Big] + \frac{H_0}{2h_l}J_{lmq}M^{-1}_{q\alpha}p_{i\alpha}\tilde{A}_{ij}M^{-1}_{m\beta}p_{j\beta}\\ &+ gh_l(\frac{1}{2}h_p M_{pl} - HI_l) - M^{-1}_{q\alpha}p_{1\alpha}N_{ql}\frac{h_l}{L_w} + \frac{1}{H_0}\partial_t R h_0 M^{-1}_{0\alpha}p_{i\alpha}I_i,\end{aligned} \tag{C.52}$$

for $\alpha, \beta \in [0, c^-]$, and the shallow-water Hamiltonian

$$\check{H}(\check{\boldsymbol{h}},\check{\boldsymbol{p}}) = \left[\frac{1}{2}h_k M^{-1}_{r\gamma}p_\gamma M^{-1}_{s\lambda}p_\lambda A_{krs} + gh_k(\frac{1}{2}h_r M_{kr} - \check{H}I_k)\right], \tag{C.53}$$

for $\gamma, \lambda \in [c^+, c+N_l]$.

C.5 Verification of the nonlinear shallow-water solutions

We verify the shallow-water solutions against the exact solution of an oscillating lake in a basin with parabolic bottom topography, as introduced *e.g.* in [19], [145] and [80]. This particular configuration enables testing the stability of the finite-volume scheme for the computations of dry regions (where non-negative depth must be ensured) and dynamic water-line boundaries. For a parabolic lake with topography

$$H(x) = H_0 \left(\frac{x}{a}\right)^2 \tag{C.54}$$

the depth-solution is described by

$$h(x,t) = H_0 - \frac{B^2}{4g} - \frac{B\omega}{g}\cos(\omega t)x - \cos(2\omega t)\frac{aB^2}{4g} - H(x), \tag{C.55}$$

where g is the gravitational acceleration, H_0 is the maximal depth at rest, ω is the frequency of the oscillations, and a and B are parameters that we set as in [145], that is

$$a = 3000, \qquad B = 5, \qquad g = 9.81, \qquad H_0 = 10 \qquad \text{and} \qquad \omega = \sqrt{2gH_0}/a. \tag{C.56}$$

Figure C.6 shows good agreement between the exact (C.55) and numerical solutions.

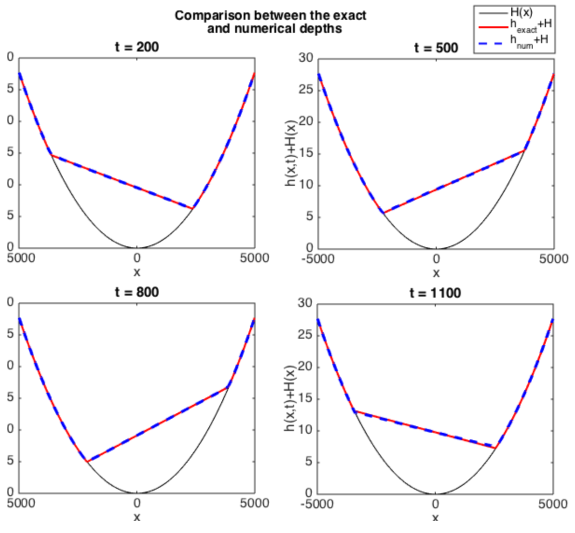


Figure C.6: Comparison between the exact (red) and numerical (blue) solutions in the case of an oscillating lake with parabolic topography (black) at different times. The numerical solution is discretised with 600 volume cells.

D Experimental validation of the numerical wave tank

D.1 Fourier modes of irregular waves

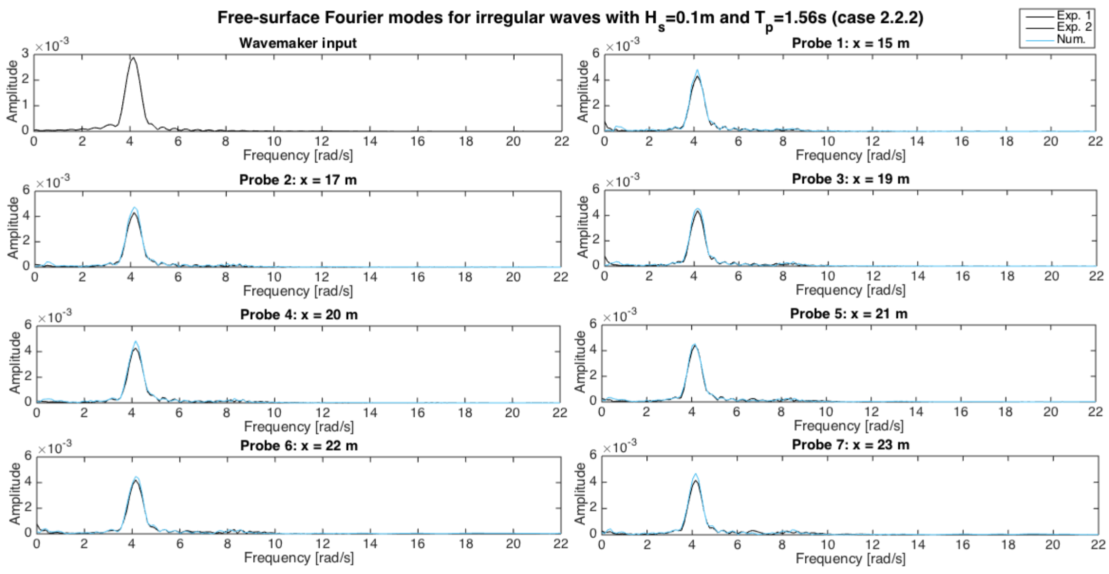


Figure D.7: Numerical (blue) and experimental (black) Fourier modes of case 2.2.2.

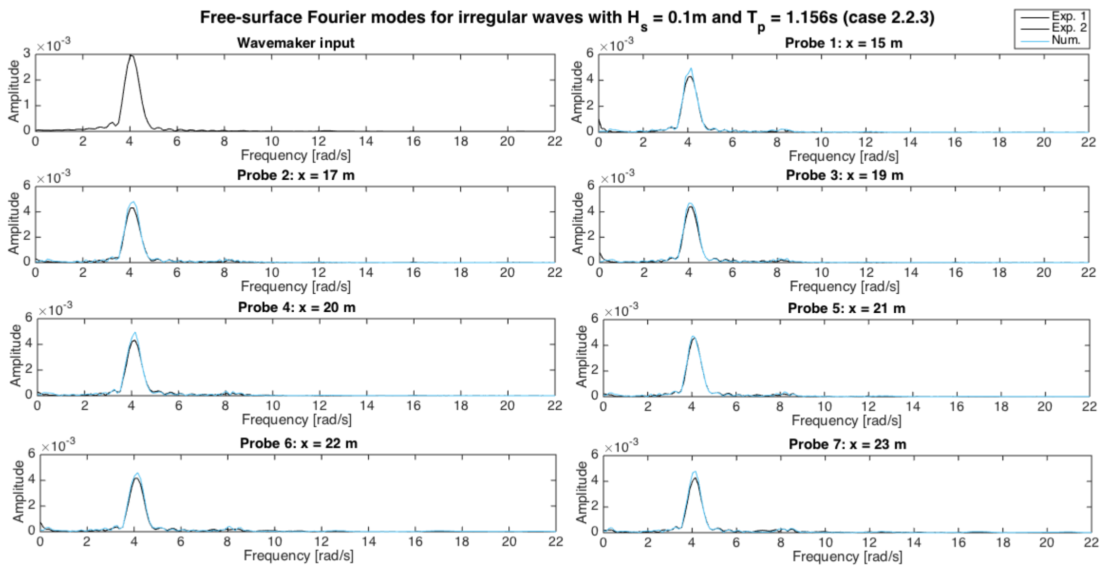


Figure D.8: Numerical (blue) and experimental (black) Fourier modes of case 2.2.3.

Bibliography

[1] M. J. Ablowitz and C. W. Curtis, *Conservation laws and web-solutions for the benney–luke equation*, Proc. Roy. Soc. A **469** (2013), 16 pp.

[2] T.A.A. Adcock, P.H. Taylor, S. Yan, Q.W. Ma, and P.A.E.M. Janssen, *Did the Draupner wave occur in a crossing sea ?*, Proc. R. Soc. A, Math., Phys. and Eng. Sci. **467** (2011), 3004–3021.

[3] J. Ahrens, B. Geveci, and C. Law, *Paraview: an end-user tool for large-data visualization*, The visualization Handbook **717** (2005), 717–731.

[4] G.B. Airy, *Tides and waves*, Encyclopaedia Metropolitana. Mixed Sciences **3** (1841).

[5] E. Audusse, F. Bouchut, M.O. Bristeau, R. Klein, and B. Perthame, *A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows*, SIAM J. of Sci. Comp. **25** (2004), 2050–2065.

[6] C. Baker, *Making a splash with solitons*, Master's thesis, University of Leeds, School of Mathematics, 2017.

[7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc users manual*, Tech. Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[8] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, and H. Zhang, *PETSc Web page*, `http://www.mcs.anl.gov/petsc`, 2017.

[9] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith, *Efficient management of*

parallelism in object oriented numerical software libraries, Modern Software Tools in Scientific Computing (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), Birkhäuser Press, 1997, pp. 163–202.

[10] BBC, *Freak wave - programme summary*, 2012.

[11] M.A. Benarde, *Our precarious habitat. an integrated approach to understanding man's effect on his environment*, W. W. Norton & Company; Second edition, 1973.

[12] D.J. Benney and J.C. Luke, *On the interactions of permanent waves of finite amplitude.*, J. Math. Phys. **43** (1964), 309–313.

[13] H. Bishop, *The night the Fitz wend down*, Duluth, Minnesota: Lake Superior Port Cities (2000).

[14] E.M. Bitner-Gregersen and O. Gramstad, *Rogue waves. impact on ships and offshore structures*, DNV GL strategic research and innovation **Position paper** (2016), no. 05-2015.

[15] E.M. Bitner-Gregersen and Ø. Hagen, *Uncertainties in data for the offshore environment*, Struct. Safety **7** (1990), 11–34.

[16] E.M. Bitner-Gregersen and A. Toffoli, *Occurence of rogue sea states and consequences for marine structures*, Ocean Dyn. **64** (2014), 1457–1468.

[17] E.M. Bitner-Gregersen and A. Toffoli, *Wave steepness and rogue waves in the changing climate in the north Atlantic*, Proc. ASME 2015 34th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2015, 2015.

[18] E.M. Bitner-Gregersen, E. Vanem, O. Gramstad, T. Hørte, O.J. Aarnes, M. Reistad, O. Breivik, A.K. Magnusson, and B. Natvig, *Climate change and safe design of ship structures*, Ocean Eng. **149** (2018), 226–237.

[19] O. Bokhove, *Flooding and drying in discontinuous Galerkin finite-element discretizations of shallow-water equations. Part 1: one dimension.*, J. of Sci. Comp. **22** (2005), no. 1-3, 47–82.

[20] O. Bokhove and A. Kalogirou, *Variational water wave modelling: from continuum to experiment*, Lectures on the theory of water waves (T.J. Bridges, M.D. Groves, and D.P. Nicholls, eds.), LMS Lecture Note Series, vol. 426, Cambridge University Press, 2016, pp. 226–260.

[21] W. Booker, T. Goodfellow, and J. Alwon, *Experimental and numerical modelling of coastal*

process, Tech. report, University of Leeds, 2015.

[22] W.J. Broad, *Rogue Giants at Sea*, New York Times (11 July 2006).

[23] B. Buchner, J. Van den Berg, J. Helder, and T. Bunnik, *Non-linear wave runup along the side of ships causing green water problems: experiments and first cfd calculations*, Proc.ASME 2014 33rd Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2014, vol. 1A: Offshore Technology, 2014.

[24] T.H.J. Bunnik, *Benchmark workshop on numerical wave modelling - description of test cases*, Tech. Report 70022-1-RD, MARIN, (2010).

[25] M.J. Castro, J.A. García-Rodriguez, J.M. González-Vida, and C. Parés, *Solving shallow-water systems in 2D domains using Finite Volume methods and multimedia SSE instructions*, J. of Comp. and App. Math. **221** (2008), 16–32.

[26] A.L. Cauchy, *Mémoires de l'académie des sciences de l'institut de france - année 1823*, ch. Mémoire sur les développements des fonctions en séries périodiques, pp. 603–612, Gauthier-Villars, 1827.

[27] L. Cavaleri, L. Bertotti, L. Torrisi, E.M. Bitner-Gregersen, M. Serio, and M. Onorato, *Rogue waves in crossing seas: the Louis Majesty accident*, J. Geophys. Res. **117** (2012), no. C00J10.

[28] A. Chabchoub, N.P. Hoffmann, and N. Akhmediev, *Rogue wave observation in a water wave tank*, Phys. Rev. Letters **106** (2011), no. 204502.

[29] J. Chambarel, C. Kharif, and O. Kimmoun, *Generation of two-dimensional steep water waves on finite depth with and without wind*, European J. of Mech. -B/Fluids **29** (2010), 132–142.

[30] G. Clauss, C. Schmittner, J. Hennig, C. Guedes Soares, N. Fonseca, and R. Pascoal, *Bending moments of an Fpso in rogue waves*, Proc. ASME 2004 23rd Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2004, 2004.

[31] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt, *Concepts and applications of finite element analysis*, John Wiley and Sons, Inc., 2007.

[32] Allianz Global Corporate and Specialty, *Safety and Shipping Review 2017. An annual review of trends and developments in shipping losses and safety*, Tech. report, Lloyd's List Intelligence Casualty Statistics, 2017.

[33] C. Cotter and O. Bokhove, *Variational water-wave model with accurate dispersion and vertical vorticity*, J. of Eng. Math. **67** (2010), 33–54.

[34] W. Craig and C. Sulem, *Numerical simulation of gravity waves*, J. Comput. Phys. **108** (1992), 73–83.

[35] A.D.D. Craik, *The origins of water-wave theory*, Annu. Rev. Fluid Mech **36** (2004), 1–28.

[36] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, *Parallel distributed computing using Python*, Advances in Water Resources **34** (2011), no. 9, 1124–1139.

[37] J.K. Devlin, *The millenium problems: The seven greatest unsolved mathematical puzzles of our time*, New York: Basic Books, 2002.

[38] I.I. Didenkulova, A.V. Slunyaev, E.N. Pelinovsky, and C. Kharif, *Freak waves in 2005*, Nat. Hazards Earth Syst. Sci. **6** (2006), 1007–1015.

[39] L. Draper, *'freak' ocean waves*, Oceanus **10** (1964).

[40] P.G. Drazin and R.S. Johnson, *Solitons, an introduction*, Press Syndicate of the University of Cambridge, 1989.

[41] K. Dysthe, H.E. Krogstard, and P. Muller, *Oceanic rogue waves*, Ann. Rev. Fluid Mech. **40** (2008), 287–310.

[42] B. Düz, M.J.A. Borsboom, A.E.P. Veldman, P.R. Wellens, and R.H.M. Huijsmans, *An absorbing boundary condition for free surface water waves*, Computers and Fluids **156** (2017), 562 – 578, Ninth International Conference on Computational Fluid Dynamics (ICCFD9).

[43] B. Düz, T. Bunnik, G. Kapsenberg, and G. Vaz, *Numerical simulation of nonlinear free surface water waves - Coupling of a potential flow solver to a URANS/VOF code*, Proc. ASME 2016 35th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2016, 2016.

[44] B. Düz, R. H. M. Huijsmans, A. E. P. Veldman, M. J. A. Borsboom, and P. R. Wellens, *An absorbing boundary condition for regular and irregular wave simulations*, MARIN 2011, IV International Conference on Computational Methods in Marine Engineering (L. Eça, E. Oñate, J. Garca, and P. Bergan an T. Kvamsdal, eds.), Springer, 2013.

[45] A.P. Engsig-Karup, H.B. Bingham, and O. Lindberg, *An efficient flexible-order model for 3D nonlinear water waves*, J. of Comp. Phys. **228** (2009), 2100–2118.

[46] Mer et Marine (J.L. Venne), *Endommagé par une vague scélérate, le jean nicoli en*

réparation à toulon, 2017.

[47] K. Fahim, *Landfall in Manhattan, after a 70-foot wave*, The New York Times (2005).

[48] Firedrake, *Solving PDEs*, 2013-2016.

[49] G.Z. Forristall, *Wave crest distributions: observations and second-order theory*, J. Phys. Oceanogr. **30** (2000), 1931–1943.

[50] M. Funakoshi, *Reflection of obliquely incident solitary waves*, J. Phys. Soc. **49** (1980), 2371–2379.

[51] E. Gagarina, A.R. Ambati, J. van der Vegt, and O. Bokhove, *Variational space-time (dis)continuous Galerkin method for nonlinear free surface water waves*, J. Com. Phys. **275** (2014), 459–483.

[52] E. Gagarina, V.R. Ambati, S. Nurijanyan, J.J.W van der Vegt, and O. Bokhove, *On variational and symplectic time integrators for Hamiltonian systems*, Journal of Computational Physics **306** (2016), 370–389.

[53] E. Gagarina, V.R. Ambati, S. Nurijanyan, J.J.W. van der Vegt, and O. Bokhove, *On variational and symplectic time integrators for Hamiltonian systems*, J. of Comp. Phys. **306** (2016), 370–389.

[54] E. Gagarina, J.J.W.van der Vegt, V.R. Ambati, and O. Bokhove, *A Hamiltonian Boussinesq model with horizontally sheared currents*, Proc. of the 3rd Int. Symp. on Shallow Flows. Iowa City: IIHR, 2012, pp. 1–10.

[55] E. Gagarina, J. van der Vegt, and O. Bokhove, *Horizontal circulation and jumps in Hamiltonian wave models*, Nonlinear Proc. in Geophys. **20** (2013), 483–500.

[56] C. Geuzaine and J.F. Remacle, *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering **79** (2009), no. 11, 1309–1331.

[57] F. Gidel, O. Bokhove, and A. Kalogirou, *Variational modelling of extreme waves through oblique interaction of solitary waves: application to mach reflection*, Nonlinear Proc. Geophys. **24** (2017), 43–60.

[58] F. Gidel, O.Bokhove, and M. Kelmanson, *Driven nonlinear potential flow with wave breaking at shallow-water beaches*, Proc. ASME 2017 36th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2017, vol. 1, 2017.

[59] N.S. Gokhale, S.S. Deshpande, and S.V. Bedekar, *Practical finite element analysis*, Finite to Infinite, 2008.

[60] O. Gramstad, H. Zeng, K. Trulsen, and G.K. Pedersen, *Freak waves in weakly nonlinear unidirectional wave trains over a sloping bottom in shallow water*, Phys. Fluids **25** (2013), no. 122103.

[61] A. Greenbaum, *Iterative methods for solving linear systems*, SIAM, Philadelphia, 1997.

[62] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration*, Springer, 2006.

[63] J. Harris, *Without Trace: The Last Voyages of Eight Ships*, Mandarin, 1989.

[64] A. Harten, P.D. Lax, and B. Van Leer, *On upstream differencing and godunov-type schemes for hyperbolic conservation laws*, SIAM Rev. **25** (1983), 35–61.

[65] S. Haver, *A possible freak wave event measured at the draupner jacket january 1 1995*, Rogue waves **460** (2004), 1–8.

[66] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, Supercomputing ’95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM) (New York), ACM Press, 1995, p. 28.

[67] J. Hennig and C.E. Schmittner, *Experimental variation of focusing wave groups for the investigation of their predictability*, Proc. ASME 2009 28th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2009, 2009.

[68] R. L. Higdon, *Absorbing boundary conditions for difference approximations to the multi-dimensional wave equation*, Math. Compu. **47** (1986), 437–459.

[69] R. L. Higdon, *Numerical absorbing boundary conditions for the wave equation*, Math. Compu. **49** (1987), 65–90.

[70] M. Hoekstra and L. Eça, *PARNASSOS: an efficient method for ship stern flow calculcation*, 3rd Osaka colloquium.

[71] M. Homolya and D.A. Ham, *A parallel edge orientation algorithm for quadrilateral meshes.*, SIAM Journal on Scientific Computing **38** (2016), S48–S61.

[72] D.V. Hutton, *Fundamentals of finite element analysis*, McGraw-Hill Higher Education, 2003.

[73] W. Irving, *A History of the Life and Voyages of Christopher Columbus*, no. vol. 1, G. & G. Carvill, 1828.

[74] N. G. Jacobsen, D. R. Fuhrman, and J. Fredsøe, *A wave generation toolbox for the open-source CFD library: OpenFoam ®*, Int. J. Numer. Meth. Fluids **70** (2012), 1073–1088.

[75] I.S.F. Jones and J.E. Jones, *Oceanography in the Days of Sail*, p. 115, Sydney Institute of Marine Science Ltd, 2088.

[76] B.B. Kadomtsev and V.I. Petviashvili, *On the stability of solitary waves in weakly dispersive media*, Sov. Phys. Dokl. **15** (1970), 539–541.

[77] A. Kalogirou and O. Bokhove, *Mathematical and numerical modelling of wave impact on wave-energy buoys*, Proc. ASME 2016 35th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2016, 2016.

[78] A. Kalogirou, O. Bokhove, and D. Ham, *Modelling of nonlinear wave-buoy dynamics using constrained variational methods*, Proc. ASME 2017 36th Int. Conf. on Ocean, Offshore and Arctic Eng., OMAE 2017, vol. 7A: Ocean Engineering, 2017.

[79] A. Kalogirou, E. E. Moulopoulou, and O. Bokhove, *Variational finite element methods for waves in a Hele-Shaw tank*, Applied Mathematical Modelling **40** (2016), 7493–7503.

[80] T. Kent, *An idealised fluid model of numerical weather prediction: dynamics and data assimilation*, Ph.D. thesis, University of Leeds, 2016.

[81] J.W. Kim and K.J. Bai, *A finite element method for two-dimensional water-wave problems*, Numerical methods in fluids **30** (1999), 105–122.

[82] F. Klaver, *Coupling of numerical models for deep and shallow water*, Master's thesis, University of Twente, Netherlands, 2009.

[83] Y. Kodama, *KP solitons in shallow water*, Journal of Physics A: Mathematical and Theoretical **43** (2010), 434–484.

[84] Y. Kodama, M. Oikawa, and H. Tsuji, *Soliton solutions of the KP equation with V-shape initial waves*, J. Phys. A: Mathematical and Theoretical **42** (2009), 312–321.

[85] D.J. Korteweg and G. de Vries, *On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves*, Philosophical Magazine **39** (1895), 422–443.

[86] W. Kristina, O. Bokhove, and E.W.C. van Groesen, *Effective coastal boundary conditions for tsunami wave run-up over sloping bathymetry*, Nonlinear Proc. in Geophys. **21** (2014), 987–1005.

[87] I.V. Lavrenov, *The wave energy concentration at the Agulhas current off South Africa*, Nat. Hazards **17** (1998), 117–127.

[88] S. Lehner, H. Gunther, and W. Rosenthal, *Extreme wave observations from radar data sets*, Ocean Waves Measurements and Analysis, 5th Int. Symp. WAVES 2005, Madrid, no. paper 69, 2005.

[89] W. Li, H. Yeh, and Y. Kodama, *On the Mach reflection of a solitary wave: revisited*, J. Fluid Mech. **672** (2011), 326–357.

[90] P.C. Liu, *A chronology of freaque wave encounters*, Geofizika **24** (2007), 57–70.

[91] M.S. Longuet-Higgins, *On the statistical distribution of the heights of sea waves*, J. Mar. Res. **11** (1952), 245–266.

[92] J.C. Luke, *A variational principle for a fluid with a free surface*, J. Fluid Mech. **27** (1967), 395–397.

[93] Q.W. Ma, G.X. Wu, and R. Eatock Taylor, *Finite element simulation of fully nonlinear interaction between vertical cylinders and steep waves. Part 1: Numerical results and validation*, Int. J. Numer. Methods Fluids **36** (2001), 265–285.

[94] Q.W. Ma, G.X. Wu, and R. Eatock Taylor, *Finite element simulation of fully nonlinear interaction between vertical cylinders and steep waves. Part 2: Methodology and numerical procedure*, Int. J. Numer. Methods Fluids **36** (2001), 287–308.

[95] Q.W. Ma and S. Yan, *Quasi ALE finite element method for nonlinear water waves*, J. of Comp. Phys. **212** (2006), 52–72.

[96] A.T.T. MacRae, G.T. Bercea, L. Mitchell, D.A. Ham, and C. J. Cotter, *Automated generation and symbolic manipulation of tensor product finite elements*, SIAM Journal on Scientific Computing **38** (2016), 25–S47.

[97] J.K. Mallory, *Abnormal waves in the south-east coast of South Africa*, Int. Hydr. Rev. **51** (1974), 89–129.

[98] MARIN, *Facilities and tools*.

[99] The Guardian (C. McGreal), *Whale-watching boat tragedy caused by freak wave, say investigators*, 2015.

[100] G.A. Meurant, *Computer solution of large linear systems*, Studies in mathematics and its applications, North-Holland ; Elsevier, 1999.

[101] A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J.K. Moore, S. Singh, T. Rathnayake, S. Vig, B.E. Granger, R.P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M.J. Curry, A.R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, *SymPy: symbolic computing in python*, PeerJ Computer Science **3** (2017), e103.

[102] V. Michel-Dansac, C. Berthon, S. Clain, and F. Foucher, *A well-balanced scheme for the shallow-water equations with topography or Manning friction*, J. of Comp. Phys. **335** (2017), 115–1554.

[103] J. W. Miles, *Obliquely interacting solitary waves*, J. Fluid Mech. **79** (1977), 157–169.

[104] J.W. Miles, *On Hamilton's principle for surface waves*, J. Fluid Mech. (1977).

[105] J.W. Miles, *Resonantly interacting solitary waves.*, J. Fluid Mech. **79** (1977), 171–179.

[106] P.A. Milewski and J.B. Keller, *Three-dimensional water waves*, Studies in Applied Mathematics **97** (1996), 149–166.

[107] N. Mori, P. Liu, and T. Yasuda, *Analysis of freak wave measurements in the sea of Japan*, Ocean Eng. **29** (2002), 1399–1414.

[108] B. Le Méhauté, *An introduction to hydrodynamics and water waves*, ch. An Introduction to Water Waves, pp. 197–211, Springer Study Edition, 1976.

[109] I. Nikolkina and I. Didenkulova, *Rogue waves in 2006-2010*, Nat. Hazards Earth Syst. Sci. **11** (2011), 2913–2924.

[110] M. Olagnon and J. Kerr, *Anatomie curieuse des vagues scélérates*, Carnets de sciences, 2015.

[111] M. Onorato, A.R. Osborne, M. Serio, L. Cavaleri, C. Brandini, and C.T. Stansberg, *Extreme waves, modulational instability and second order theory: wave flume experiments on irregular waves*, European J. of Mech. - B/Fluids **25** (2006), 586–601, Rogue waves European Geosciences Union Assembly.

[112] M. Onorato, D. Proment, and A. Toffoli, *Triggering rogue waves in opposing currents*, Phys. Rev. Lett. **107** (2011), no. 184502.

[113] R. L. Pego and J. R. Quintero, *Two-dimensional solitary waves for a Benney-Luke equation*, Physica D **132** (1999), 476–496.

[114] D.H. Peregrine, *Water waves, nonlinear Schrödinger equations and their solutions*, J.

Austral. Math. Soc. Ser. B **25** (1983), 16–43.

[115] R. Perić and M. Abdel-Maksoud, *Reliable damping of free-surface waves in numerical simulations*, Ship Technology Research **63** (2016), no. 1, 1–13.

[116] R. Perić and M. Abdel-Maksoud, *Analytical prediction of reflection coefficients for wave absorbing layers in flow simulations of regular free-surface waves*, Ocean Engineering **147** (2018), 132–147.

[117] P.H. Perroud, *The solitary wave reflection along a straight vertical wall at oblique incidence.*, Tech. report, University of California, Berkley, 1957.

[118] P. Peterson, T. Soomere, J. Engelbrecht, and E. van Groesen, *Soliton interaction as a possible model for extreme waves in shallow water*, Nonlinear Processes in Geophysics **10** (2003), 503–510.

[119] L.J. Pratt, *On inertial flow over topography, Part 1: Semigeostrophic adjustment to an obstacle*, J. Fluid Mech. **131** (1983), 195–218.

[120] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, G. Bercea, G. R. Markall, and P. H. J. Kelly, *Firedrake: automating the finite element method by composing abstractions*, 2016.

[121] J. N. Reddy, *An introduction to the Finite Element Method*, McGraw-Hill, 2005.

[122] O. Reynolds, *On the rate of progression of groups of waves and the rate at which energy is transmitted by waves*, Nature **16** (1877), 343–344.

[123] W. Rosenthal and S. Lehner, *Rogue waves: results of the MaxWave project*, J. Offshore Mech. Arct. Eng. **130** (2008), 21006–21013.

[124] W. Rosenthal, S. Lehner, H. Dankert, H. Guenther, K. Hessner, J. Horstmann, A. Niermeier, J.C. Nieto-Borger, J. Schulz-Stellenfleth, and K. Reichert, *Detection of extreme single waves and wave statistics*, Rogue Waves: Forecast and Impact on Marine Structures, GKSS Research Center, Geesthacht, Germany (2003).

[125] J.S. Russell, *Report on waves*, Report of the fourteenth meeting of the british association for the adcancement of science (1844), 311–390.

[126] Y. Saad, *Iterative methods for sparse linear systems*, 2nd. ed., SIAM, 2003.

[127] T. Salwa, O. Bokhove, and M.A. Kelmanson, *Variational modelling of wave-structure interactions with an offshore wind-turbine mast*, J. of Eng. Math. **107** (2017), 61–85.

[128] A. Sergeeva, E. Pelinovsky, and T. Talipova, *Nonlinear random wave field in shallow water: variable Korteweg-De Vries framework*, Nat. Hazards Earth Syst. Sci. **11** (2011), 323–330.

[129] C.B. Smith, *Extreme waves.*, Washington, D.C.: Joseph Henry Press (2066), 68–69.

[130] G. D. Smith, *Numerical solution of partial differential equations: Finite-difference methods*, 2nd. ed., p. 217, Clarendon Press, Oxford, 1978.

[131] G.G. Stokes, *On the theory of oscillatory waves*, Transactions of the Cambridge Philosophical Society **8** (1847), 441–455.

[132] G.G. Stokes, *Supplement to a paper on the theory of oscillatory waves*, Cambridge University Press **1** (1880), 214–326.

[133] M. Tanaka, *Mach reflection of a large-amplitude solitary wave.*, J. Fluid Mech. **248** (1993), 637–661.

[134] M.A. Tayfun, *Narrow-band nonlinear sea waves*, J. Geophys. Res. **85** (1980), 1548–1552.

[135] A. Toffoli, E.M. Bitner-Gregersen, A.R. Osborne, M. Serio, J. Monbaliu, and M. Onorato, *Extreme waves in random crossing seas: Laboratory experiments and numerical simulations*, Geophys. Res. Lett. **38** (2011), no. L06605.

[136] A. Toffoli, L. Cavaleri, A.V. Babanin, M. Benoit, E.M. Bitner-Gregersen, J. Monbaliu, M. Onorato, A.R. Osbone, and C.T. Stansberg, *Occurence of extreme waves in three-dimensional mechanically generated wave fields propagating over an oblique current*, Nat. Hazards Earth Syst. Sci. **11** (2011), 895–903.

[137] A. Toffoli, L. Fernandez, J. Monbaliu, M. Benoit, E. Gagnaire-renou, J.M. Lefévre, L. Cavaleri, D. Proment, C. Pakozdi, C.T. Stansberg, T. Waseda, and M. Onorato, *Experimental evidence of the modulation of a plane wave to oblique perturbations and generation of rogue waves in finite water depth*, Phys. Fluids **25** (2013), no. 091701.

[138] K. Trulsen, J.C. Nieto Borge, O. Gramstad, L. Aouf, and J.M. Lefévre, *Crossing sea state and rogue wave probability during the Prestige accident*, J. Geophys. Res.: Oceans **120** (2015), 7113–7136.

[139] K. Trulsen, H. Zeng, and O. Gramstad, *Laboratory evidence of freak waves provoked by non-uniform bathymetry*, Phys. Fluids **24** (2012), no. 097101.

[140] G. Vaz, F. Jaouen, and M. Hoekstra, *Free-surface viscous flow computations. validation of URANS code FreSCo*, Proc. ASME 2009 28th Int. Conf. on Ocean, Offshore and Arctic

Eng., OMAE 2009, 2009.

[141] E.H. Weber and W.E. Weber, *Wellenlehre auf experimente gegründet*, Leipzig: Gerhardt Fleischer, 1825.

[142] W. Weibull, *A statistical distribution function of wide applicability*, J. Appl. Mech. **18** (1951), 293–297.

[143] H. G. Weller and G. Tabor, *A tensorial approach to computational continuum mechanics using object-oriented techniques*, Computers in Physics **12** (1998), 620.

[144] G.X. Wu and Z.Z. Hu, *Simulation of nonlinear interactions between waves and floating bodies through a finite-element-based numerical tank*, Proc. R. Soc. Lond. Ser. A **460** (2004), 2797–2817.

[145] Y. Xing, X. Zhang, and C. Shu, *Positivity-preserving high order well- balanced discontinuous galerkin methods for the shallow water equations*, Advances in Water Resources **33** (2010), no. 12, 1476–1493.

[146] H. Yeh, W. Li, and Y. Kodama, *Mach Reflection and KP Solitons in Shallow Water*, European Physical Journal **85** (2010), 97–111.