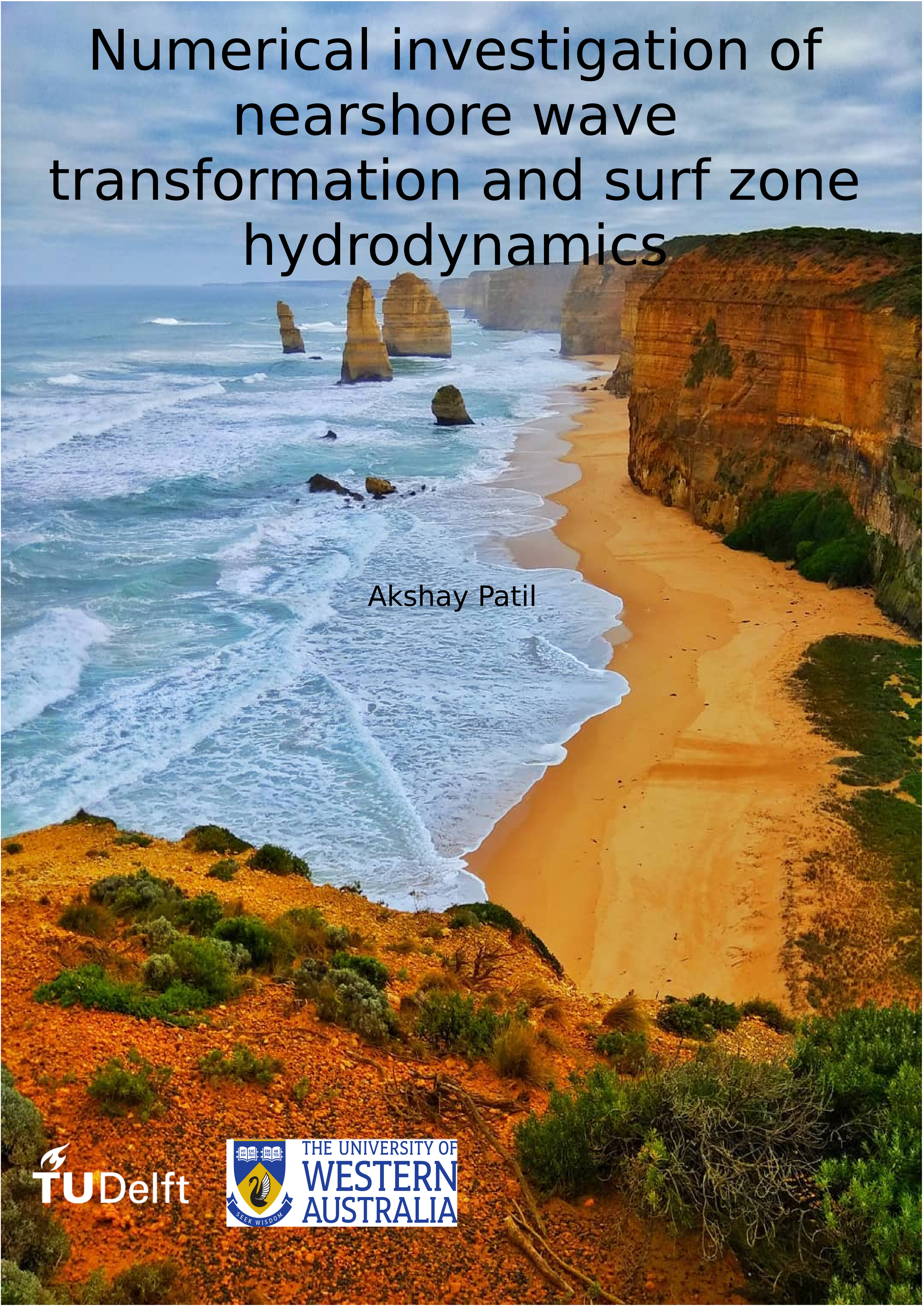
Numerical investigation of nearshore wave transformation and surf zone hydrodynamics



Akshay Patil

[This page intentionally left blank]

Numerical investigation of nearshore wave transformation and surf zone hydrodynamics

By

Akshay Patil

in partial fulfilment of the requirements for the degree of

Master of Science
in Hydraulic Engineering

at the Delft University of Technology

Student Number	4655745	
Thesis committee	Prof. Dr. Ir. Ad Reniers	TU Delft
	Dr. Ir. Jeremy Bricker	TU Delft
	Prof. Dr. Ryan Lowe	University of Western Australia
	Dr. Niels Jacobsen	Deltares
	Ir. Chris Lashley	TU Delft
	Ir. Patrick Oosterlo	TU Delft

[This page intentionally left blank]

ABSTRACT

Rapid climate change and the corresponding estimated sea level rise can affect the performance of the coastal defense structures such as breakwaters, seawalls, and dikes. In order to improve these coastal defenses, a detailed understanding of the processes which contribute to wave run-up and overtopping over the coastal defenses needs to be established. Following the exponential growth of computing capacity around 1970's, a wide variety of computational models were developed to study fluid flow. Traditionally, three computational paradigms have existed in order to study wave transformation and surf zone hydrodynamics: phase averaged models, phase resolving models, and Computational Fluid Dynamics (CFD) models. Limitations posed by the underlying linear wave theory in phase averaged and other simplifications in the phase resolving models, may not provide sufficient detail in wave breaking, wave energy dissipation, wave run-up, wave overtopping, and potentially other detailed hydrodynamic processes. This lack of resolution in depth-averaged models for wave-breaking, wave run-up, and wave overtopping processes motivates a detailed investigation using CFD based models, which can correctly mimic wave-breaking and other hydrodynamic processes.

The recent growth in available computational capacity has greatly improved the applicability of CFD based models for large scale transient flows such as waves near a coast. Additionally, the developments in wave generation and wave absorption boundary conditions by Jacobsen et al. [2012] in the open-source CFD toolbox OpenFOAM ®, have facilitated the use of OpenFOAM in coastal engineering applications. This encourages investigating the coastal environment using relatively complex models, thus providing insights into fundamental processes which contribute to coastal safety. To that end, this thesis focuses on investigating wave overtopping and the underlying processes which contribute to the aforementioned hydrodynamic aspects.

Overtopping demands accurate capture of the free surface (interface between water and air). The *waveFoam* solver suffers from numerical diffusion of the interface, consequently requiring a different approach to mimic the sharp interface. In order to cater to this deficiency, a new solver which combines the capabilities of waveFoam [Jacobsen et al., 2012] and isoAdvection [Røenby et al., 2016] which has the ability to capture sharp interfaces by means of a sub-grid approach has been integrated (*waveFlow*) and used in this study. In addition to the new solver, a new set of Reynolds Averaged Navier-Stokes (RANS) closures developed by Larsen and Fuhrman [2018] for wave modeling applications have been employed to correctly capture turbulence levels under breaking waves. The preliminary steps include calibrating and assessment of the newly integrated *waveFlow* solver. Using a relatively simple conceptual test case, a comparison of the free surface behavior and overtopping discharge was carried out. This calibration test was followed by a comparison of numerical results with the experimental investigations carried out by Ting and Kirby [1994]. Following this benchmarking study, experimental studies carried out by Flanders Hydraulics investigating wave overtopping over dikes in shallow foreshore environments was validated. A comparison of *waveFlow* and *waveFoam* was made to assess the qualitative and quantitative differences between the two interface capture methods on overtopping. Using this new solver, OpenFOAM was able to reproduce the surface elevation and significant improvement in the overtopping results were obtained for identical model setup in comparison to the *waveFoam* solver. A coupled approach using a potential flow solver named OceanWave3D aided simulation of large domain wave propagation and helped to cut down the computational time.

[This page intentionally left blank]

ACKNOWLEDGEMENTS

This report is the result of the numerical investigations carried out over a period of about 7 months. Personally, it has been one of the most challenging projects that I have worked on. I can formally acknowledge that the lessons learnt during this academic endeavour will definitely help me in the coming years.

First I would like to thank my family and loved ones for all the love and support they have extended for the past two years. Your constant encouragement and care has definitely made it a fair bit easier to stay focused and complete the MSc program here in Delft. I would like to thank Mruga for her unconditional love and motivation during the entire two years. A lot of things would not have been possible without your encouragement and wise words. I would also like to thank Prasad for his never-ending motivation and support. I would like to extend my thanks to all my friends here in Delft and elsewhere for the help and company during the MSc journey. Special thanks to all my friends in Delft for making this experience of living abroad such a wonderful one, I will fondly remember all the wonderful memories we shared together in the past two years.

Personally and professionally it has been an exhilarating experience to work with what I think has been one of the best research committee. I wish to thank all my mentors for providing the opportunity to work in such a vibrant and rewarding research area of Environmental Fluid Mechanics. I would like to extend my special thanks to Ad for chairing the thesis committee and providing essential feedback on the research project and for the critical review on the numerical aspects. I would like to thank Jeremy for all the support, encouragement, practical advise, and recommendations during the past one year of intense academic research work where we have been involved. I would also like to thank Ryan for including me in this project and for the extensive feedback and mentoring. In addition to this, I want to thank Ryan for inviting me to the beautiful city of Perth to collaborate with his colleagues from the University of Wester Australia (UWA). I want to thank Chris for all of the wonderful feedback and patience during the discussion sessions and weekly meetings, I have indeed learnt a lot about research from you. I would like to extend a special thanks to Niels who has provided me the necessary motivation and support during the entire project duration. Thank you for all the wonderful discussions about numerics and hydrodynamics, you have certainly inspired me to be a better researcher. I would like to thank Patrick for his extensive comments on my writing and analysis of the data, I appreciate your patience and support.

I would like to thank Lina for the detailed discussions about turbulence modeling and wall functions which have greatly helped me get a better insight. I also want to thank all the wonderful people in 3.92 for the pleasant working environment. I want to thank Arkajit for his help with C++ and python. I want to extend my thanks to Johan Rønby and Bjarke Larsen for some detailed discussion via E-mail and for taking out the time to provide additional comments on the inner working on their developments within OpenFOAM. I would like to thank the HPC08 computing cluster and its users. I also want to formally acknowledge the support provided by PAWSEY, Perth for the computing facilities provided, this thesis work would not have been possible without PAWSEY. I also want to thank Alexis for all the HPC support at PAWSEY.

Akshay Patil
Düsseldorf, August 2019

[This page intentionally left blank]

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

OUTLINE

This chapters aims to introduce the main area of investigation. A brief overview of the problem will first be presented, this will be followed by the research outline which discusses the research objective along with the research questions that will be addressed in this thesis. The successive sections will discuss the scope and methodology of this thesis work.

1.1 BRIEF OVERVIEW

Low-lying countries such as Belgium and the Netherlands have a coastal defense system along the coastal urban areas. Some of these protected coastal areas are characterized by a mildly sloped and very shallow beach in front of a dike [Vandebeek et al., 2018] (See Figure 1.1). Sea level rise and extreme climatic events can considerably question the current coastal defense installations in such low-lying countries. In order to avoid the possible damage and provide adequate safety, most of the affected regions are reinforcing the sea defenses. Strengthening such coastal defenses requires a detailed understanding of the underlying processes which result in wave run-up and overtopping in such shallow coastal environments.

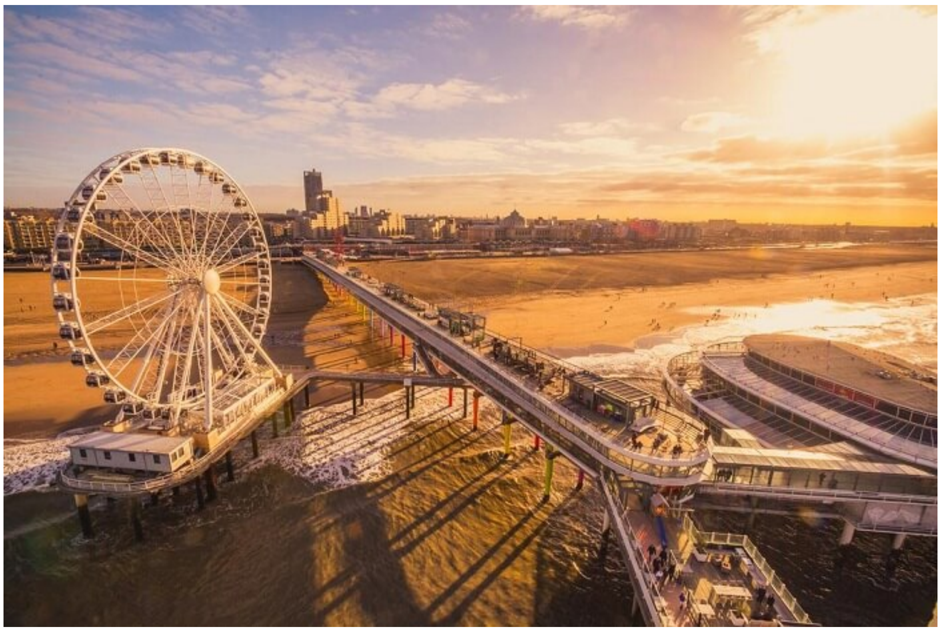


Figure 1.1: Scheveningen beach (Den Haag, Netherlands) depicting a shallow coast with high-rise buildings running along the coast.

In the past (before the catastrophic flood of 1953 in the Netherlands), wave run-up was considered to be one of the most important parameters for determining the height of the coastal defense. However, since then overtopping volume along with run-up has been found out to be a crucial factor which determines the integrity of the coastal defense system [Schiereck, 2016]. This enforces the accurate prediction of overtopping volume and wave run-up over a sloping beach with a coastal defense. In the past, the surf-zone has been studied considerably, both experimentally and numerically. A wide range of geometries for regular and irregular waves have been investigated mainly dealing with the *mean* overtopping discharge [Stansby and Feng, 2004]. In this region of interest (surf zone), the incident offshore waves are substantially transformed as a result of the underlying bathymetry and other associated non-linear processes. One of the most important phenomena which occur near the coast is wave-breaking. Wave-breaking is seen as one of the last and most complex life events of a wave. As the waves move closer to the shoreline, the wave height increases due to shoaling until the wave cannot stay vertical anymore resulting in forward pitching and breaking. This breaking results in the generation of high amounts of turbulence, which is one of the key governing processes converting organized wave motion into turbulence. As quantification of the turbulence based on field observations is not always possible, a large number of experimental studies have been conducted e.g., Ting and Kirby [1994], Beij and Battjes [1993], Svendsen and Putrevu [1995], [Devolder et al., 2018]. As opposed to a simplified experimental approach, the practical conditions under which such coastal defenses exist are rather complex, waves are irregular, directional, oblique and generally propagate over immensely complex bathymetry. This hinders any kind of empirical studies which require restricted defining criteria, thus demanding a modeling scheme which is able to process the above-defined input conditions [Stansby and Feng, 2004].

Since prototype/field observations are not always possible and scaled physical models may have constraints with regards to replicating given wave conditions in most cases, numerical wave models can provide a link to reproduce field conditions subject to certain approximations/simplifications. One of the central considerations in designing a numerical wave model for practical usage, is limiting the computational time. This computational limitation is posed by the Courant criterion which dictates that the wave energy ($c_{g,x}$ in this case) may not travel more than one geographic cell in one time step [Holthuijsen, 2007]. This implies that $\Delta t < \frac{\Delta x}{c_{g,x}}$, where Δt is the numerical time step [s], Δx is the numerical grid size [m], and $c_{g,x}$ is the group wave velocity in the propagation direction [m/s]. Another way to look at the Courant condition (commonly known as Courant–Friedrichs–Lewy [CFL] condition) is to follow figure 1.2. The red line in the figure depict the region of dependence for the grid point under consideration (marked yellow). For oceanic waters, the value of $\Delta x \sim$ 25-100 km, this gives a Δt between 20-80 min for the lowest frequency of waves $\sim$ 0.04 Hz. In coastal waters, the value of $\Delta x \sim$ 10-100 m, while the lowest frequency stays the same. This results in a $\Delta t \sim$ 1.5-15 s, thus rendering the model operationally unacceptable [Holthuijsen, 2007].

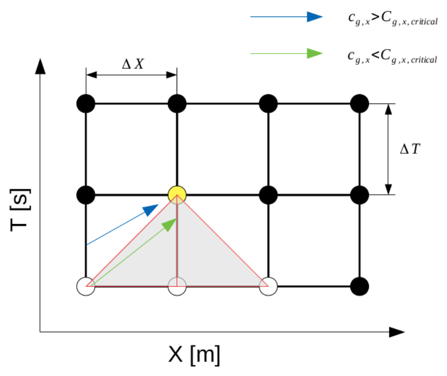


Figure 1.2: Graphical representation of the CFL condition

Motivated by the lack of computational power and a need to model such coastal systems, a wide variety of models were introduced. Traditionally, three computational paradigms have existed in order to study wave transformation and surf zone hydrodynamics: phase averaged models, phase resolving[1] models, and Computational Fluid Dynamics (CFD) models. A schematic of the various classes of numerical models has been presented in Figure 1.3.

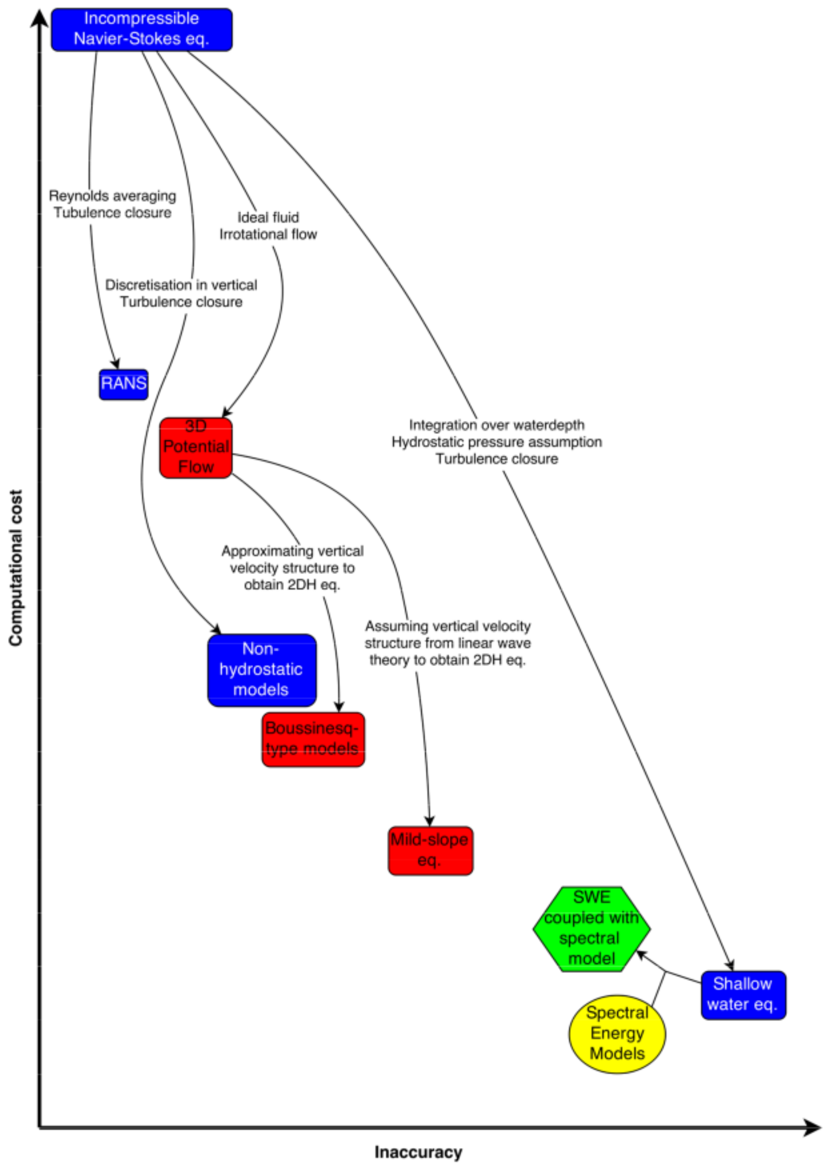

Incompressible Navier-Stokes eq.
Reynolds averaging Tubulence closure
Ideal fluid Irrotational flow
Discretisation in vertical Turbulence closure
RANS
3D Potential Flow
Integration over waterdepth Hydrostatic pressure assumption Turbulence closure
Approximating vertical velocity structure to obtain 2DH eq.
Assuming vertical velocity structure from linear wave theory to obtain 2DH eq.
Non-hydrostatic models
Boussinesq-type models
Mild-slope eq.
SWE coupled with spectral model
Spectral Energy Models
Shallow water eq.
Computational cost
Inaccuracy

Figure 1.3: Various computational models positioned on the computing time vs level of accuracy graph [van Mierlo, 2014].

In this thesis, the focus is mainly going to be on the left section of Figure 1.3 i.e., the Incompressible Navier-Stokes equations with Reynolds Averaged Navier-Stokes turbulence closure. The rise in highly parallelized computing system and advances in computing power have made it possible to investigate wave hydrodynamics using small scale turbulence resolving models like the one considered in this thesis. This thesis builds upon the work carried out previously by Jacobsen et al. [2012], Larsen

[1]In this context, these are models which do not resolve multiple interfaces.

and Fuhrman [2018], Røenby et al. [2016], and Zhou et al. [2014] with regards to wave modeling and multiphase flows.

1.2 RESEARCH OUTLINE

Research Objective

In order to investigate wave hydrodynamics and provide a better prediction in the required hydrodynamic parameters contributing to wave overtopping, a detailed modeling strategy is required. To that end, this thesis aims to answer a few relevant questions with regards to wave hydrodynamics. The principal focus of this thesis is to understand the hydrodynamic processes involved in wave overtopping over coastal structures by integrating a CFD model (`waveFlow`) capable of reproducing the hydrodynamics including wave-breaking in shallow foreshore environments .

Research Questions

In order to achieve the previously mentioned objective a systematic research strategy has been formulated. This strategy aims to answer a central question which seeks to gain insight into the physics of wave hydrodynamics by means of a numerical model.

What is the practical feasibility of advanced CFD based models in comparison to simpler models like phase averaged and phase-resolving models, in complex surf zones and swash zones?, and which physical processes in the surf zone affect wave overtopping on the slopes of coastal structures like dikes?

This research questions sets the theme for a set of detailed sub-questions to answer the principal research question and thus achieve the research objective.

Sub-Questions

- *What hydrodynamics of the surf-zone can be obtained with an advanced model employing a RANS based approach?*
- *How is the turbulent kinetic energy distributed for spilling and plunging wave breaker using a RANS based approach?*
- *What additional improvement can be observed in the prediction of wave overtopping over coastal dikes using a RANS based approach in comparison to simpler models?*
- *What physical processes contribute to wave overtopping and how well are they replicated in the numerical model?*

1.3 SCOPE AND METHODOLOGY

Scope

The area of investigation as described in section 1.2 is multifaceted and involves a synergy of different fields of science (SeeFigure 1.4). In addition to this, the area of wave hydrodynamics is quite broad, as a result, it is important to specify the area of focus in this particular thesis.

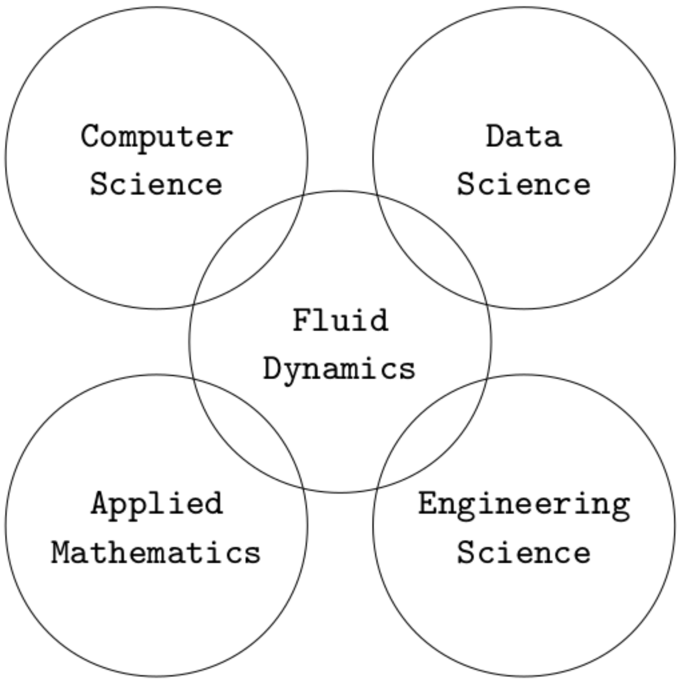


Figure 1.4: Different fields of science involved in order to investigate the hydrodynamics of waves near a coastal structure.

This thesis will focus on the following basic tasks:

1. Integrate `waves2Foam` and `isoAdvection` into a new solver named `waveFlow` within the `OpenFOAM` framework and assess its performance in comparison to the old solver originally developed by Jacobsen et al. [2012]. This new solver will be assessed for wave overtopping performance.
2. Investigate turbulence characteristics using a RANS based approach with `waveFlow`. This study will also permit a feasibility investigation to use RANS based turbulence closures for relatively long term wave modeling applications.
3. Investigate the physical processes contributing to wave overtopping over coastal defense structures like dikes using a numerical model and assess the capabilities of the numerical model.
4. Address the additional benefits derived by using an advanced numerical model and turbulence closure and also list any shortcomings in the modeling strategy.

Methodology

Large scale experimental studies have already been carried out in order to investigate swash and surf zone hydrodynamics see, Beij and Battjes [1993], Ting and Kirby [1994], Scott et al. [2009]. In this thesis, a numerical model will be used to gain extensive insights into the surf and swash zone hydrodynamic processes. The numerical investigations involve a few steps as described below:

- Integrate a new solver within the `OpenFOAM` environment based on the previous studies carried out by Jacobsen et al. [2012] and Røenby et al. [2016].
- Carry out a proof of concept test case for the newly developed solver.

- Benchmark the numerical model for ***monochromatic waves*** with spilling and plunging wave breakers.
  - Carry out a comparative analysis of the experimental results and numerical results.
  - Outline the shortcomings and difficulties in the numerical model. Arrive at a guesstimate for the error bar involved with using the numerical model in the current state.
- Investigate numerical model performance for ***irregular waves*** for wave overtopping over a coastal dike.
  - Qualitative and quantitative comparison of `waveFoam` and `waveFlow` solvers with respect to overtopping.
  - Assess the performance of the numerical model against the experimental data and outline the strengths and shortcomings of the numerical model.

1.4 OUTLINE OF THE THESIS

The subsequent chapters will consist of an extensive literature review in Chapter 2 which provides the state of the art description for coastal hydrodynamics and the numerical advances made within the domain of computational hydraulics. This chapter will also address some of the difficulties associated with modeling waves using various computational approaches. In Chapter 3, the development of the new solver `waveFlow` and a conceptual test case will be discussed to highlight the differences between the two interface capture methods tested in this thesis. Chapter 4 presents the model setups and domain descriptions for the two experimental investigations replicated. This chapter also provides some insights into the different data analysis routines used in the thesis work. Chapter 5 discusses in depth the results for both the experimental campaigns replicated in the numerical model. This chapter also aims to provide a concise yet complete description of the results in terms of the various variables compared in the experiments and the numerical model. In the concluding chapter the research questions proposed in Chapter 1 will be re-addressed in new light and the final recommendations will be provided. The rest of the text in the thesis consists of the appendices which hosts most of the C++ and python code used to carry out the analysis.

2 LITERATURE REVIEW

OUTLINE

In this chapter an overview of the established knowledge about surf and swash zone dynamics will be presented. Each section focuses on a different aspect and aims to provide a thorough but condensed perspective on the corresponding scientific advancements. Section 2.1 details the hydrodynamic processes in the surf and the swash zone. In section 2.2, the fluid instabilities which eventually lead to turbulence will be addressed without restriction to shallow waters. In the following section, the turbulence under breaking waves will be addressed. Section 2.4 discusses the empirical understanding of the overtopping processes. The final section of this chapter details the computational aspects of modeling coastal environments.

2.1 SURF AND SWASH ZONE HYDRODYNAMICS

The coastal zone can be divided into a number of different sections. A schematic of this classification can be seen in Figure 2.1. Most waves are either generated due to the effect of wind (generally termed as short waves) or due to storms[1]. Such events generally introduce energy into the oceanic water system. This energy is introduced as a result of the momentum exchange between wind and water or pressure gradients created due to storms which eventually generate surface curvature and drive the flow. This energy propagates in the form of waves across the ocean reaching the coastal regions. Since the propagation of waves in deeper oceanic water is different from shallower waters like those found in the coastal zones (see Holthuijsen [2007]), complex dynamics emerges in the behavior of the waves due to this interaction with the sea surface. It is in this surf and swash zone that one of the most interesting and complex life-stages of a wave occurs i.e., wave breaking. This is where the incoming wave energy is transformed into highly chaotic turbulent structures and dissipated eventually in the form of heat.

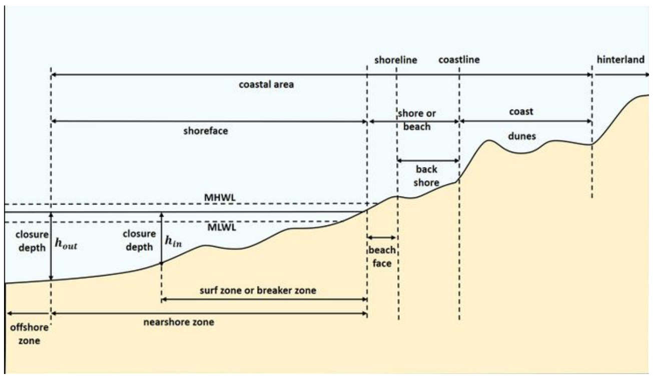


Figure 2.1: Cross-sectional view of the coastal region depicting the various zones. Adapted from Coastal Engineering Research Council [1984].

[1]Other wave generation mechanisms exist, however they are not listed here.

Although the above description of wave generation is limited to wind generated waves, a wide variety of waves coexist in the oceanic system. A spectral overview of the different types of waves can be seen in Figure 2.2. Additionally, there are a wide variety of hydrodynamic and morphodynamic processes which can be observed in the coastal zones other than wave breaking itself. Processes like cross-shore, along-shore currents, morphodynamic response of the beach, density currents, stratification, etc., although have interesting dynamics, will be excluded from the description for sake of brevity. In this and the following sections, the description of waves in oceanic system will be restricted to wind waves only. Considering a large amount of energy exists in this part of the energy spectrum, it will be of central focus.

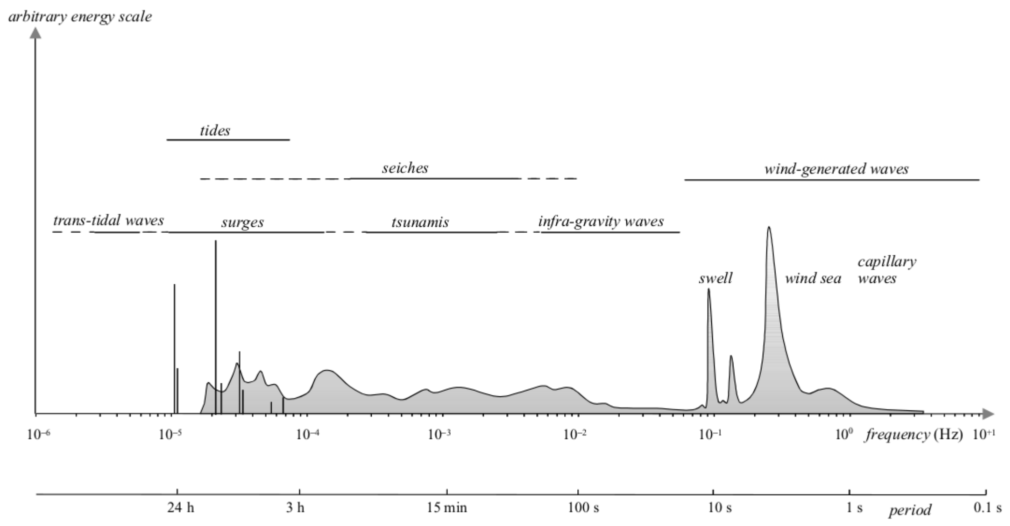


Figure 2.2: Frequencies and periods of the vertical motions of the ocean surface [Holthuijsen, 2007]

According to Arcilla and Lemos [1990], the relevant processes in the surf zone can be classified in roughly four categories:

- Sediment transport and corresponding changes in morphology, with a characteristic time scale of 1 day to 1 month, and a spatial scale between 100 m and 1000 m,
- Currents (non-oscillatory flow), with time scales between 10 minutes and 1 hour, and spatial scales similar to those of sediment transport,
- Organized oscillatory flows (i.e., wind waves, infra-gravity waves), with time scales ranging from 10^{-1} sec to 10 min, and space scales from 1 to 100 m.
- Random oscillatory flow (turbulence), whose length scales are between 10^{-3} to 10^{1} sec , and with small (10^{-4} to 10^{-1} m) spatial scales.

2.2 WAVE BREAKING

2.2.1 Deep water wave breaking

Despite the early proof of the presence of Stokes (regular) type waves, which were initially proposed by Stokes [1847] and later confirmed by Benjamin [1967], the growth and presence of instabilities in periodic stoke wave propagation was unknown/uninvestigated until C Yuen and M Lake [1980] showed the existence of deep water wave instabilities. The seminal work by S Longuet-Higgins and D. Cokelet

[1976], investigating the evolution of the Benjamin-Feir instabilities first proposed by T. Benjamin and J. Feir [C Yuen and M Lake, 1980] describes the development of modulation (sideband) instabilities. These modulation instabilities are deviations from a periodic wave form propagation which are reinforced by the non-linearity (advection term in this case) in the governing partial differential equation. For long term wave propagation, the spectral-sideband eventually breaks up into train of pulses [Benjamin and Feir, 1967]. As seen in Figure 2.3, the power spectrum shows a clear side band spectrum for an AM broadcast signal[2]. Although, wave propagation is different, it bears such resemblances with other energy propagation systems.

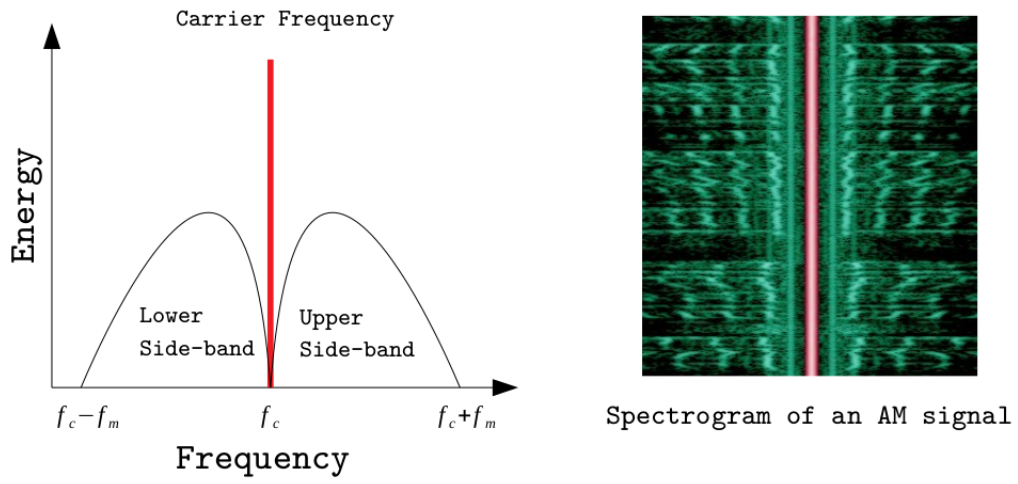


Figure 2.3: The power spectrum of a typical side banded signal. In this figure the carrier frequency (f_c) is the main component at which the AM broadcast is traveling, while the sidebands represent the transmitted modulation. Here f_m corresponds to the maximum modulation frequency. The right side of the figure also depicts the spectrogram of such a spectrum (AM broadcast) clearly depicting the sideband (green shades) and the carrier frequency (in red)

Theoretical investigation of such instabilities have restrictions for a variety of reasons, some of them have been listed below:

- Non-linearity of the surface boundary condition (air-water interface)
- Unsteady nature of the flow
- Breaking maybe a transitional process, thus bordering the limits between laminar and turbulent flow. The latter having mathematically unsolved questions by definition [Melville, 1982].

Consequently a wide variety of numerical investigations pertaining the evolution and growth of such instabilities leading to wave breaking have been carried out by Yuen and Ferguson [1978], Fuhrman et al. [2006], Chalikov [2007], and Kharif and Touboul [2010] to name a few. Since the main focus of this study is to investigate the effect of wave breaking in the shallow surf and swash zone, only the presence of the deep water instabilities is acknowledged without going into extensive details. The next sub-section describes the formulations and description of wave breaking in the shallow coastal environment.

[2]Image Source: Wikipedia.

2.2.2 Shallow water wave breaking

The term "wave breaking" (See Figure[3] 2.4) in this section is described as the transition from a smooth wave to the quasi-steady state with a white-water front at any particular instant within the transition [Peregrine, 1983]. Such wave breaking can occur both in deep and shallow water regimes of the wave life cycle. Although the resulting wave breaking looks similar to that observed in deep waters, the processes leading to this state are vastly different. The flow instabilities which lead to wave breaking in deep waters have been discussed in section 2.2.1. The wave breaking process as described by Peregrine [1983] in general is easier to observe around any coastal region where the waves break due to structural gravity based processes.

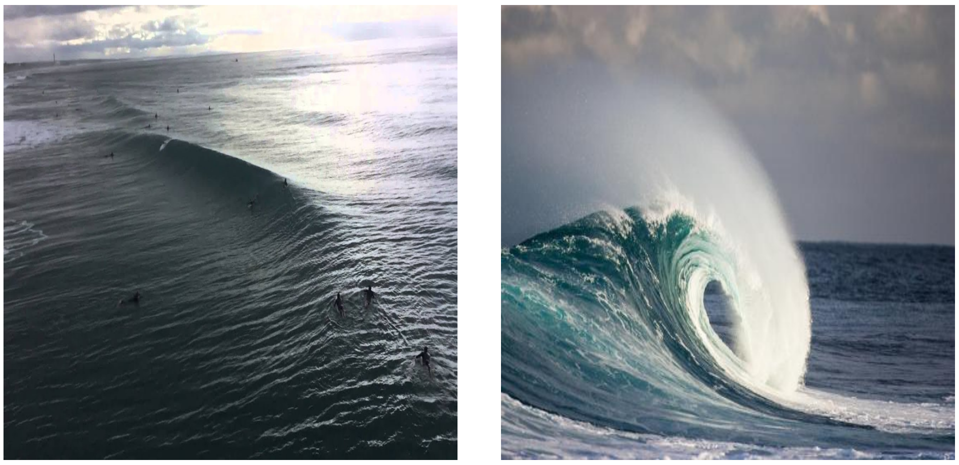

(a) Shoaling Wave (b) Breaking Wave

Figure 2.4: The figures above show a distinct contrast in the two stages of the wave life cycle. The term wave breaking describes this transition from shoaling wave to foam featured breaking wave.

According to Battjes [1988], the theoretical criterion for the onset of periodic waves breaking over a slope (beach), can be regarded as the limiting conditions for solutions representing non-breaking standing waves. This generalization is based on the previous studies carried out by Carrier and Greenspan [1958]. They derived an exact standing-wave solution for inviscid, nonlinear, shallow-water equations. The solutions derived in this study hold while the parameters $\epsilon_w < 1$ defined as given in equation 2.1.

$$\epsilon_w = \frac{\omega^2 a_w}{g\alpha^2} \tag{2.1}$$

Where ϵ_w is the breaking parameter, ω is the wave period $[rad/s]$, a_w is one half of the total vertical waterline excursion $[m]$, g is the gravitational acceleration $[m/s^2]$, and α^2 is the beach slope (after using small angle approximation $sin(\alpha) \approx \alpha$).

The local surface slope stability was evaluated to be critical when the value for $\epsilon_{w,cr} = 1$. This means that the surface becomes vertical (locally) when this value is reached. Munk and Wimbush [1969] derived the criterion by attributing ϵ_w as a parameter which is the ratio of the maximum downslope acceleration ($\frac{\omega^2 a_w}{\alpha^2}$) and downslope component of gravitational acceleration ($g\alpha$). Additional investigations for the linear shallow water equations were carried out following these studies. Linear solutions valid for arbitrary depths were also introduced by Keller [1963], Miche

[3]Source: https://www.art.com/products/p45909342294-sa-i10492339/jefffarsai-wave-breaking-in-ocean.htm

[1944], and Pocklington [1921][4]. The solutions to the linear shallow water equations predict an amplification factor $\frac{a_w}{2a_0} = (\frac{\pi}{2\alpha})^{1/2}$, in which $2a_0$ is the amplitude at an anti-node of the standing wave in deep water. In order to transform the incident wave conditions to those of deep water, the parameter a_w was replaced by $2a_0$ while ignoring the slope dependant amplification [Battjes, 1988].

The above descriptions for wave breaking criterion are based on analysis of the non-linear and linear shallow water equations which are derived by simplifying the Navier-Stokes equations which govern fluid flow. In addition to the analytical expressions, empirical expressions relating the physical properties to wave breaking were also investigated in depth by Iribarren and Nogales [1949] which were later modified by Battjes [1974].

As discussed by Svendsen and Putrevu [1995], the surf similarity parameter ζ is defined as per Equation 2.2,

$$\zeta = \frac{h_x}{\sqrt{H/L_0}}, \tag{2.2}$$

where L_0 is the deep water wavelength, h_x is the bottom slope, and H is the deep water wave height. This parameter was derived for the situation of standing waves on a steep beach with full reflection as opposed to waves breaking over a gently sloping beach. An underlying assumption which is not clearly reflected in the above formulation of the surf similarity parameter is that, the waves break at the first node from the shoreline. The depth at this node is then used as the characteristic depth which is in-turn used to determine the breaking wave height. Although this parameter has garnered a good reputation to describe the surf zone wave conditions, it does not completely resemble the wave motion in the actual surf-zone. One explanation on why the surf-similarity parameter seems to describe the surf-zone conditions so well could be ascribed to the beach slope parameter. Since there is a definitive relationship between the beach slope parameter and the surf similarity parameter as described in Equation 2.3, it could explain why the surf similarity can predict surf zone wave breaking conditions [Svendsen and Putrevu, 1995].

$$S_b = 2.3\zeta_0 \tag{2.3}$$

2.3 TURBULENCE CHARACTERISTICS UNDER BREAKING WAVES

The previous section discussed the routes which lead to wave breaking, eventually cascading into turbulent structures within the fluid column in the surf zone. To summarize the previous section, surf zone turbulence originates from instabilities of surface waves which lead to the release/conversion of kinetic energy by means of wave breaking [Serio and Mossa, 2006]. Breaking waves and the associated flow field govern the velocity distribution of a large number of hydrodynamic/morphodynamic parameters [Christensen and Deigaard, 2001]. The extremely unsteady and non-uniform nature associated with breaking waves in the surf zone has been evident since the pioneering work by Stive [1980]. This section will detail the available knowledge on the turbulence characteristics under breaking waves without any restrictions to regular wave conditions.

[4]The results are usually ascribed to Miche [1944], however, they were also derived by Pocklington [1921] earlier.

2.3.1 Overview of turbulence in waves

The need to investigate the temporal and spatial distribution of k (Turbulent Kinetic Energy) stems from a fundamental need to model the coastal environment. Since antiquity a wide range of turbulence approximations (closures) have been utilized to incorporate the effect of fluctuating velocity components in the flow domain. Simple models such as, the eddy viscosity model which follows equation 2.4 showed sufficient capability to model wave flows well [Svendsen, 1987]. In addition to simple algebraic models which aimed at describing the flow details, advanced models like the two equation $k - \epsilon$ or $k - \omega$ (ϵ is TKE dissipation rate while ω is specific TKE dissipation rate) based models also included a general transport equation for the relevant quantities. Since the determination of k is fundamental to evaluate the turbulent diffusion coefficient, and consequently, the turbulent mass transport (see equation 2.5), a wide variety of studies have investigated the turbulence characteristics within the surf zone [Serio and Mossa, 2006].

$$\nu_t \propto k^{1/2} \tag{2.4}$$

where ν_t is the eddy viscosity and k is the turbulent kinetic energy.

$$\frac{\partial k}{\partial t} + \frac{\partial \langle u_j \rangle k}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{1}{\rho}\langle u_j^{'} p^{'} \rangle + \langle u_j^{'} k^{'} \rangle - 2\nu \langle u_j^{'} s_{ij}^{'} \rangle\right) - \langle u_i^{'} u_j^{'} \rangle \frac{1}{2}\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i}\right) - 2\nu \langle s_{ij}^{'} s_{ij}^{'} \rangle \tag{2.5}$$

where u_j represents the velocity component, k represents the turbulent kinetic energy, x_j is the spatial coordinate, ρ is the fluid density, $k^{'}$ is the fluctuating component of the turbulent kinetic energy, and s_{ij} is the strain-rate tensor. To that end, the studies carried out by Svendsen [1987] analyzed the distribution of turbulence kinetic energy (k) in the vertical and horizontal coordinates. One of the crucial observations in this study was the spreading of turbulence towards the bottom and a simultaneous decrease in the intensity. In the surf zone where the waves are characterized by turbulent bore like structures emerging from wave breaking on the incoming waves, the processes involved are schematized in Figure 2.5.

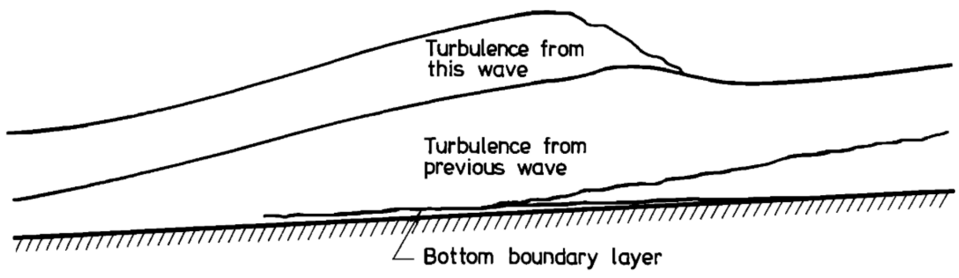


Figure 2.5: Turbulence region under broken waves [Svendsen, 1987]

Figure 2.5 also brings to light the split between the water column, the time-averaged velocity field in the surf-zone is shoreward above the through level while and undertow in the offshore direction. The time averaged velocity profile in the surf zone can be seen in figure 2.6. This velocity profile results in a high shear layer in the central region (due to large velocity gradient) which tends to be unstable. Before the wave breaks, periodic undulations of the vortex surface are produced which resemble the Kelvin-Helmholz (K-H) instabilities. As a result of these instabilities vortical structures emerge in the water column [Watanabe et al., 2005]. The experiments carried out by Li and Dalrymple [1998] also investigated the instabilities and the resulting vortex structure. One such vortex within the fluid domain can be seen in figure 2.7 which represents the general picture under the breaking waves.

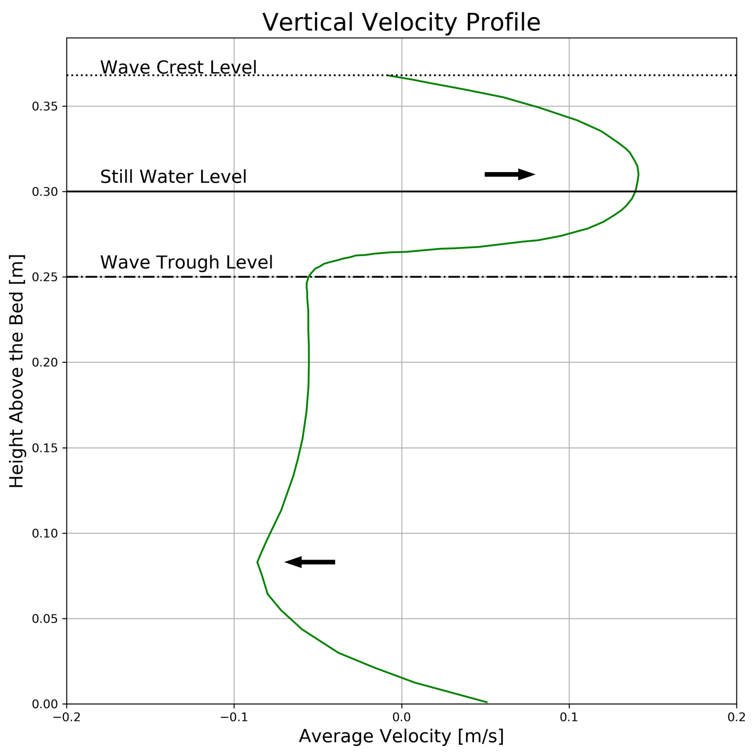


Figure 2.6: Time-averaged velocity profile in the surf zone. Adapted from [Bosboom and Stive, 2011]

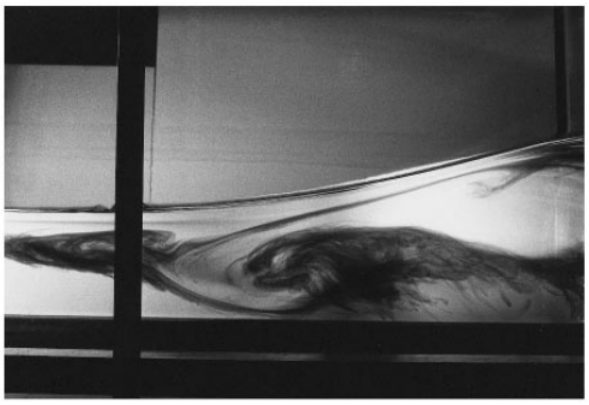
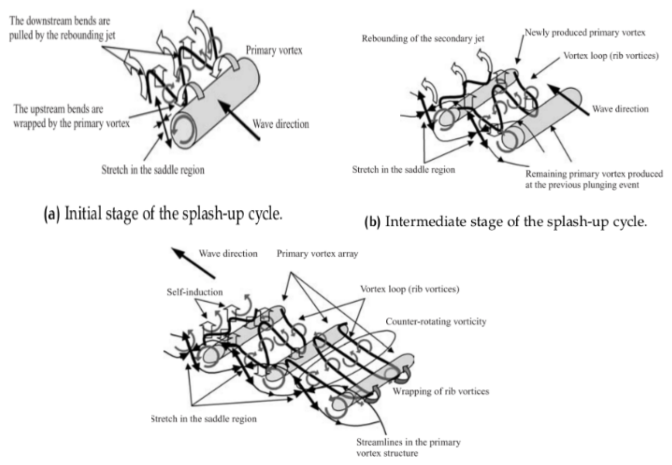


Figure 2.7: Dye pattern depicting vortex structure beneath the wave trough [Li and Dalrymple, 1998]. The wave propagates to the left side in this figure.

2.3.2 Obliquely descending eddies (ODE)

As discussed in the previous section, there are two primary mechanisms by which wave breaking is achieved. Since the central area of interest is the surf-zone, the role of shallow water wave breaking is considered in this section, the details have been discussed in section 2.2.2.

The overturning jet in the plunging breaker generally produce a 2-D horizontal rollers which disintegrates into 3-D vortical structures [Zhou et al., 2017]. Watanabe et al. [2005] also showed that the splash-up[5] cycles in both spilling and plunging breakers result in such initial horizontal rollers/eddies/bores. However, the strength of the roller is lower in the spilling breaker given that the front face merely spills down thus initiating the conversion of kinetic energy in the system. In comparison to the spilling breaker, the plunging breaker has higher kinetic energy thus the resulting splash-up cycle is largely chaotic and results in a stronger roller. The strength of the rollers decreases with successive splash-up which represents a propagating bore [Watanabe et al., 2005]. The development of such a roller can be seen in Figures 2.8 and 2.9.


The downstream bends are pulled by the rebounding jet
Primary vortex
The upstream bends are wrapped by the primary vortex
Wave direction
Stretch in the saddle region
Rebounding of the secondary jet
Newly produced primary vortex
Vortex loop (rib vortices)
Wave direction
Stretch in the saddle region
Remaining primary vortex produced at the previous plunging event
Wave direction
Primary vortex array
Self-induction
Vortex loop (rib vortices)
Counter-rotating vorticity
Wrapping of rib vortices
Stretch in the saddle region
Streamlines in the primary vortex structure

(a) Initial stage of the splash-up cycle.

(b) Intermediate stage of the splash-up cycle.

(c) Developed stage of the splash-up cycle.

Figure 2.8: Schematic of the evolution of 2-D horizontal rollers into 3-D vortical structures [Watanabe et al., 2005].

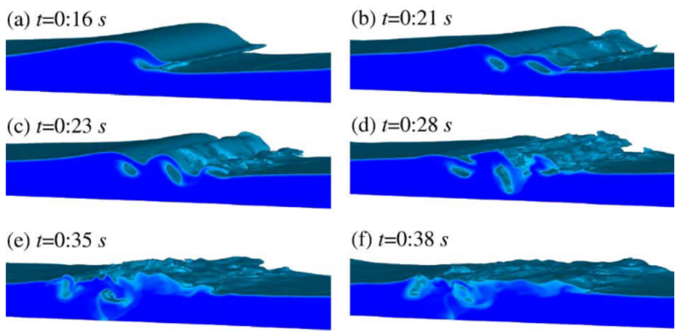


Figure 2.9: Numerical simulation of a plunging wave at different times [Lubin et al., 2006].

The turbulent kinetic energy is transported shoreward and vertically downward in the spilling breaker and seaward and vertically downward in the plunging breaker. The rate of turbulent energy diffusion in the spilling breaker is slower than that in the plunging breaker. This has been attributed to the presence of strong large eddies in the plunging breaker which help to transport energy effectively [Serio and Mossa, 2006]. Presence of these vortical structures in breaking waves has been detailed in some of the experimental work by Svendsen [1987], Ting and Kirby [1994], and Scott et al. [2009] to name a few. In addition to the experimental inves-

[5]Splash-up cycles result in the horizontal eddies

tigations some numerical studies by Zhou et al. [2014], Ozdemir et al. [2013], and Zhou et al. [2017] to list a few, have also provided an insight into the presence of vortical structures within breaking waves.

RANS closures assume an isotropic turbulence field, as a result, the cascade process as seen in Figure 2.10 is only partially captured by the RANS closure. While the turbulence scales greater than the grid size are resolved, the sub-grid processes are modeled by means of a turbulent eddy viscosity, TKE production and dissipation terms. As a result, the 3D cascade processes as discussed in the previous section does not require explicit resolution. The sub-grid model handles the right production and dissipation terms which try and mimic the right cascade behavior. Consequently, using a RANS approach is valid in this case even when one is using a 2-D modeling approach (the current study).

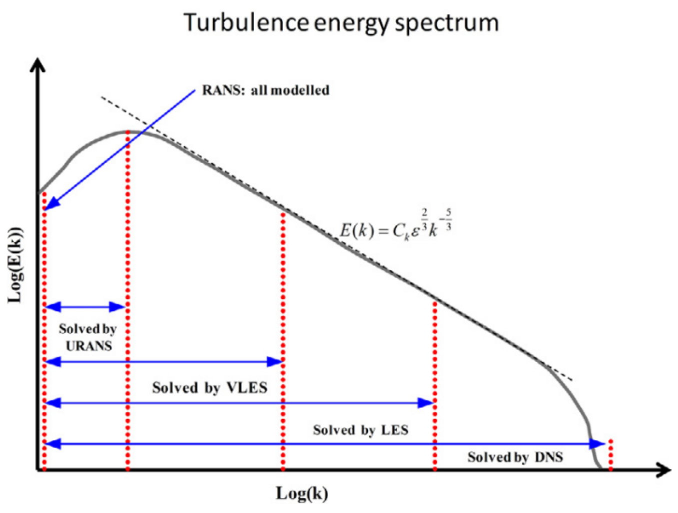


Figure 2.10: Resolving turbulence cascade process by different computational models, adapted from [Argyropoulos and Markatos, 2015].

2.4 WAVE OVERTOPPING

2.4.1 Traditional approach

A wide variety of experimental investigations have been carried to study the surf zone waves. The technical advisory committee on flood defense (Netherlands) has produced a series of technical reports titled "Wave Run-up and Wave overtopping at Dikes" (Taken from the translated version, Original version titled "Golfoploop en golfoverslag" published starting from the year 1972). The governing parameters for the determination of the height of the dike (See Figure 2.11) are listed as follows [Van der Meer, 2002] (See TAW 1999-2 for details for Sea and Lake)[6]:

- The reference level with a probability of being exceeded corresponding to the legal standard;
- The high water increase or lake level increase during the design period;
- The expected local ground subsidence during the design period;
- The bonus due to squalls, gusts, seiches and other local wind conditions;

[6]This manual has been superseded by the EurOtop manual which can be found at `http://www.overtopping-manual.com/`

- The expected decrease in crest height due to settling of the dike body and the under soil during the design period;
- The wave run-up height and the wave overtopping height.

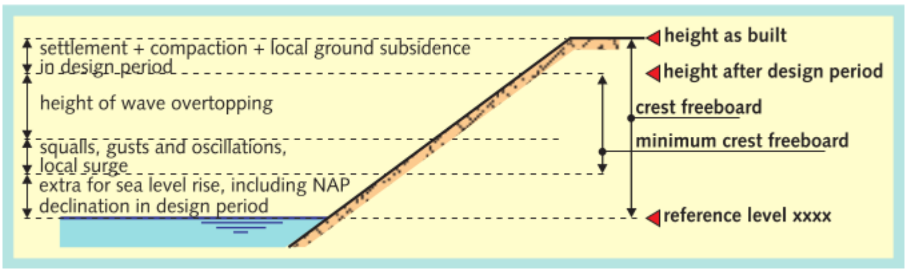

settlement + compaction + local ground subsidence in design period
height of wave overtopping
squalls, gusts and oscillations, local surge
extra for sea level rise, including NAP declination in design period
height as built
height after design period
crest freeboard
minimum crest freeboard
reference level xxxx

Figure 2.11: Important elements of a dike governing the dike height [Van der Meer, 2002]

Following this framework, a number of empirical formulations have been developed for estimation of wave run-up. The report discusses a wide variety of geometries and wave incidence conditions. The classification of the wave run-up data is based on the breaker parameter as defined in Equation 2.6.

$$\zeta_0 = \frac{tan(\alpha)}{\sqrt{s_0}} \tag{2.6}$$

where (See Figure 2.11) ζ_0 is the breaker parameter, α is the angle of slope, S_0 is wave steepness = $2\pi H_{m0}$ / (g $T^2_{m-1,0}$), H_{m0} is the wave height ($4\sqrt{m_0}$), $T_{m-1,0}$ is the spectral wave period = m_1/m_0, m_n correspond to the n^{th} moment of the wave spectrum, and g is the acceleration due to gravity. This definition of the surf similarity parameter uses the spectral description for incident wave conditions. Using such a breaker parameter, a general formulation for wave run-up can be made (See Van der Meer [2002] for details). Introducing additional correction terms to cater for various simplifications can be made such as listed below:

- Effect of slope
- Influence of shallow foreshore
- Influence of angle of incidence of wave attack
- Influence of berm
- Influence of berm width r_B
- Influence of roughness elements
- Influence of vertical or very steep wall or a slope

Such correction factors can accommodate for the effects due to individual elements present in the dike and foreshore (See Van der Meer [2002] for definition) system. Similar analytical/empirical formulation can be made for the mean/average wave overtopping discharge. The general measure to obtain the severity of wave overtopping is by means of and *'average discharge per linear meter of width'* q $[m^3/s/m]$. To summarize, similar assumptions/corrections factors can be included for wave overtopping related estimation.

The outer slope is generally protected using filter layers and bed protection, it is the inner slope that is susceptible to substantial damage due to wave overtopping. The guidelines suggest the following average discharges are indicative of erosion of the inner slope [Van der Meer, 2002]:

- 0.1 [$l/s/m$] for sandy soil with a poor grass cover;
- 1.0 [$l/s/m$] for clayey soil with a reasonably good grass cover;
- 10 [$l/s/m$] for clay covering and a grass cover as per requirements.

The text above focused on wave overtopping from a perspective which culminates in an acceptable safety based on the average overtopping discharge. However, Studies by Taylor and Williams [2019], Whittaker et al. [2018], Hofland et al. [2014] and Tromans et al. [1991] have portrayed the shift from an average description of coastal safety to an extreme wave condition based approach (See section 2.4.2). Technical reports similar to the Van der Meer [2002] were also published by the Coastal Engineering Research Center, Army Corps of Engineers, Mississippi, USA [Coastal Engineering Research Council, 1984]. The EurOtop manual is the state of the art manual which has been well received by the coastal engineering community [EurOtop et al., 2018]. However, for sake of brevity the contents will not be discussed in this report and reference to the report has been made for the readers interested.

For design purposes a variety of empirical formulations for wave overtopping have been developed (See EurOtop et al. [2018], Altomare et al. [2016], Hofland et al. [2015]). Since such empirical formulations heavily rely on statistical measures for quantification of run-up and overtopping, the number of waves required to arrive at a satisfactory estimate is quite large ($\sim$ 1000 waves). The requirement for the number of waves is motivated by the statistical characterization of the random sea-state where about 20-30 min's (1200-1800 s) are generally used [Holthuijsen, 2007]. Assuming a wave period of $\sim$ 3 s, gives roughly $\sim$ 500 waves. Consequently, most of the above studies use an average/mean descriptor for evaluating wave overtopping discharge. Despite the effectiveness of the design formulas based on this approach, the requirement of $>$ 1000 waves limits the use of physical model and numerical models for long term wave applications. Hence, the use of numerical models (Navier-Stokes or RANS approach) has been historically limited to short time durations. As a result, numerical investigations pertaining wave overtopping and wave run-up using mean overtopping discharge for $\sim$ 1000 waves has not been realized yet.

2.4.2 *NewWave* approach

Most coastal dike designs follow a process as sketched in Figure 2.12, it is critical to note that in the design stage the most extreme event (anticipated) is used as the design storm [Loffredo et al., 2007]. Using this information, the time domain of the incident wave conditions can be reduced to manageable levels.

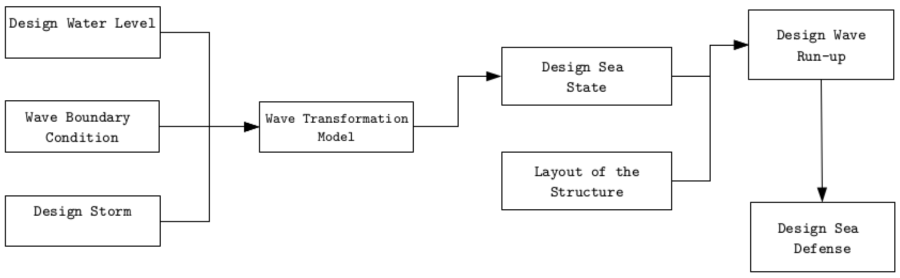


Figure 2.12: Conventional coastal dike design approach as a schematic (adapted from [Loffredo et al., 2007]).

Omitting the waves that result in no or significantly small overtopping volumes helps in reducing the time duration for which the physical and numerical models are employed. Presently, arriving at an extreme response includes the use of $\sim$ 1000 waves for a standard JONSWAP spectrum [Hofland et al., 2014]. This leads to a storm duration of about 2-3 hours at the prototype scale and about 35-40 mins at the lab scale. Such time durations provide sufficient statistical accuracy, but are almost entirely non-feasible when using numerical models for detailed investigation. Given that the extreme response is generated by a single wave group, replacing the design sea state (DSS) with a substitute sea state (SSS) can provide an effective means to estimate the extreme incident conditions for the design of coastal dikes. This substitute sea state can be derived from the original incident wave conditions by means of a systematic analysis [Hofland et al., 2014].

The wave overtopping analysis carried out by Hofland et al. [2014] investigated the effect of two different approaches for reducing the time domain of the incident wave conditions:

- Simple Sampling
- New Wave Approach

In order to test the two different ways to analyze long time series they first investigated the reproducibility of the incident wave conditions of slightly different coastal dike structures. The premise for this approach is that the same wave group that causes an extreme event in the time window (say) Δt_1 for structure 1 also results in an extreme event within the time window (say) Δt_2 for structure 2 which has only been slightly altered. They present (See Figure 2.13) normalized cumulative overtopping volumes for different dike configurations and confirm that for slightly modified (roughness values) structures the same wave groups result in extreme wave conditions. They also acknowledge the differences obtained for different configurations especially in terms of exceedance probabilities. Thus meaning that different wave group leading to the maximum overtopping in one configurations could lead to 5^{th} largest overtopping event in another configuration. Since no individual overtopping analysis was done, it was not possible to distinguish the individual events. Thus, it was concluded that wave groups leading to an extreme event in one configurations also resulted in an extreme event in a different configuration, however with a different exceedance probability (rank number). Given that the design is governed by these extreme events, this seems to be a feasible approach over simulating the entire storm event.

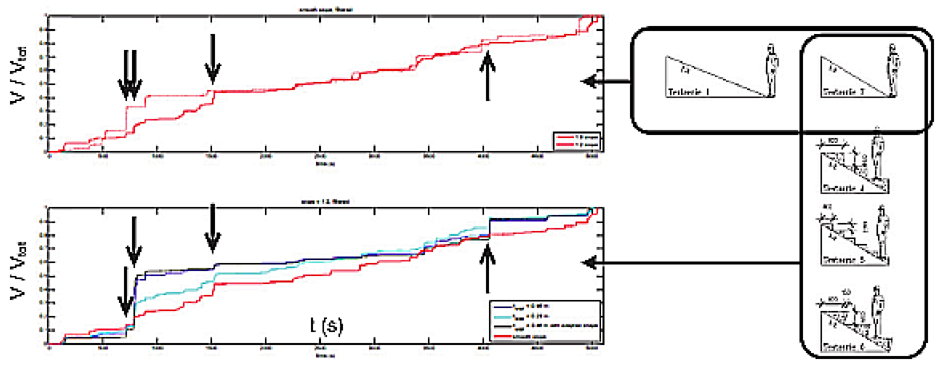


Figure 2.13: Left: normalized overtopping volume for different structures with the same wave field. Right: corresponding structures. Top: smooth 1:2 and 1:3 slopes. Bottom: different geometry and size of roughness elements on outer slope (Taken from Hofland et al. [2014]).

Following this investigation, the two approaches mentioned in the preceding section were investigated.

Simple sampling

Using the original DSS a modified version of this time series was created. A sensitivity analysis for the length of the modified signal was carried out. Slicing and comparing different time durations of the original DSS provided with a measure for the effect of the corresponding sampled time duration on the overtopping volumes recorded. It was found out that the duration may be linked to the time taken by the smaller waves to reach the structure given as

$$T_{slow} = \int_L 1/c_g(2f_p)dx \tag{2.7}$$

as seen in Figure 2.14, T_{slow} is indicated by the vertical line in the figure. There seems to be some sort of correlation with the amount of reflection present for each case. For example, case W1 shows that about $1T_{slow}$ is required for adequate resolution, this case has low reflection in the system. At the same time, for case W2 steepest waves were incident and consequently largest reflection in comparison. This also resulted in a larger time series about $\sim 3-4T_{slow}$ to adequately resolve wave overtopping. However, it wasn't clear why there was a large discrepancy in the W1 overtopping values even for relatively large values of sampled time series.

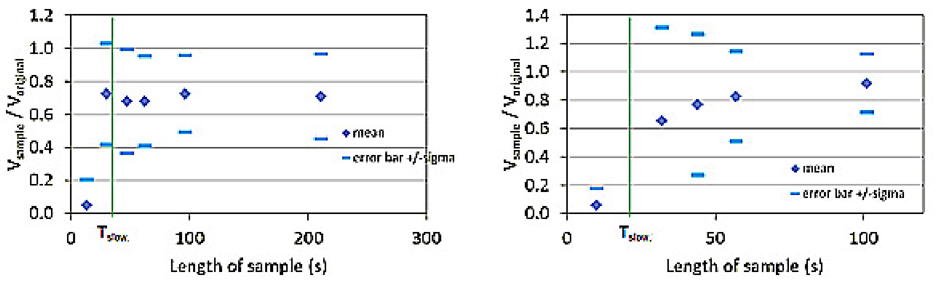


Figure 2.14: Normalized overtopping volume (ratio of overtopping volume in sampled and in original test), as function of sample length, T s . Left: case W1, right: case W2. (Taken from Hofland et al. [2014]).

New wave approach

In cases where the wave height is the most important parameter that determines wave overtopping, then it is expected that the tenth highest wave heights would on average result in the tenth largest overtopping volume. As shown by Tromans et al. [1991], a representative wave height with a given exceedance probability would immediately results in the corresponding overtopping event. For cases where design is governed by the extreme event, this proves to be a suitable candidate for limiting the time duration of the incident wave conditions. In this study, Hofland et al. [2014] investigated $H_{13\%}$, $H_{1\%}$, and $H_{0.1\%}$ respectively. It was shown that the New Wave approach yields an overtopping volume which is roughly within a factor of 2 in comparison to the 10,000 waves investigated in the experimental investigations. This can be seen in Figure 2.15.

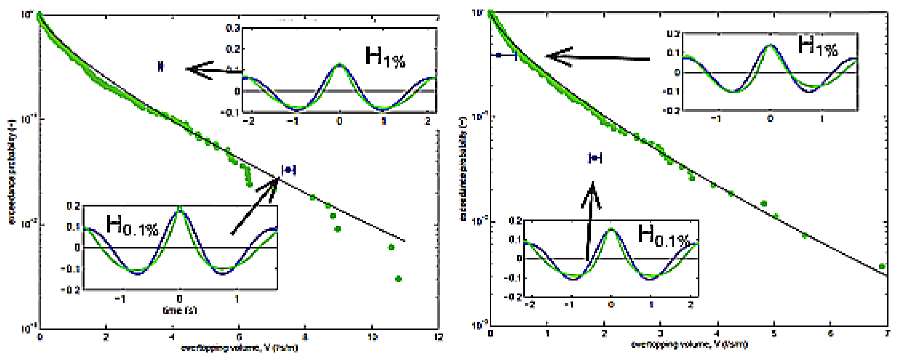


Figure 2.15: Overtopping by New Wave (blue) compared to extreme value distribution of random DSS. Left: case W1, right: case W2. Inserts: theoretical (blue) and realized (green) surface elevation of New Waves at the toe of the dike (Taken from Hofland et al. [2014]).

As detailed in the studies mentioned above, the efficacy of this approach could help alleviate the trouble in simulating long time series as required for reliable overtopping predictions. Despite the promising results a detailed investigation is required to establish the effectivity of this approach for modeling wave overtopping using a small time domain. In addition to this, the uncertainty in the approach has not been commented upon in terms of wave overtopping, which could be significant given that only one data point is obtained from a given wave group [Hofland et al., 2014]. In addition to this, the effect of low frequency waves, wave setup, and many more hydrodynamics cannot be readily distinguished thus limiting the scope of this approach. Consequently, this approach still requires additional investigations and will not be considered in this thesis work. However, the author does acknowledge that isolating the shortcomings in this approach might aid the use of this approach for reducing the long time series replication in the numerical model.

2.5 MODELING THE COASTAL ENVIRONMENT

The dynamic processes present in the coastal environment are a result of a widely different driving agents. The characteristic motions occur at various timescales ranging from turbulent small scale features to the scale of currents and tides. Consequently modeling such a system would require involving all the necessary agents to be included in order to solve engineering problems concerning the coastal environment [Brocchini and Dodd, 2008]. Some of the physical processes and the corresponding scales have been listed and discussed in section 2.1. In the following section, a concise discussion about the various type of models for inspecting the coastal zone will be carried out.

2.5.1 Phase Averaged Models

Phase averaged models are models which do not distinguish between individual waves but instead use a prognostic variable like the wave spectrum to predict wave transformation. Since the wave spectrum is of interest in this case, these models are also known as 'spectral models'. The governing equations for phase-averaged models can either be wave-energy balance or wave-action balance. In this particular section, the wave-energy balance based model will be presented to qualitatively differentiate this type of model.

A basic energy balance equation governs the spectral model, this can be summarized in equation 2.8. The underlying assumption for a phase averaged model is the existence of random phase/amplitude model which contributes to the energy of the spectrum. It is this wave energy spectrum that is transported in a given geographic domain over time. Now in order to predict the wave spectrum at a given location, the wave energy spectrum at the point of inception (say the coast) has to be integrated with correct accounting for all the source and sink terms of energy from the coast till the prediction point [Holthuijsen, 2007]. A simplified schematic detailing this methodology can be seen in Figure 2.16. For sake of brevity only the relevant bits are discussed in this chapter.

$$\frac{\partial E(\omega,\theta;x,y,t)}{\partial t} + \frac{\partial c_{g,x} E(\omega,\theta;x,y,t)}{\partial x} + \frac{\partial c_{g,y} E(\omega,\theta;x,y,t)}{\partial y} = S(\omega,\theta;x,y,t) \quad (2.8)$$

In equation 2.8, $E(\omega,\theta;x,y,t)$ is the energy density spectrum, $c_{g,x}$ is the propagation velocity in x-direction, $c_{g,y}$ is the propagation velocity in y-direction, and $S(\omega,\theta;x,y,t)$ can include all dissipation and generation mechanisms. Equation 2.8 works with the absence of any ambient current, i.e., no frequency shifting. Detailed treatment and discussions can be seen in Holthuijsen [2007].

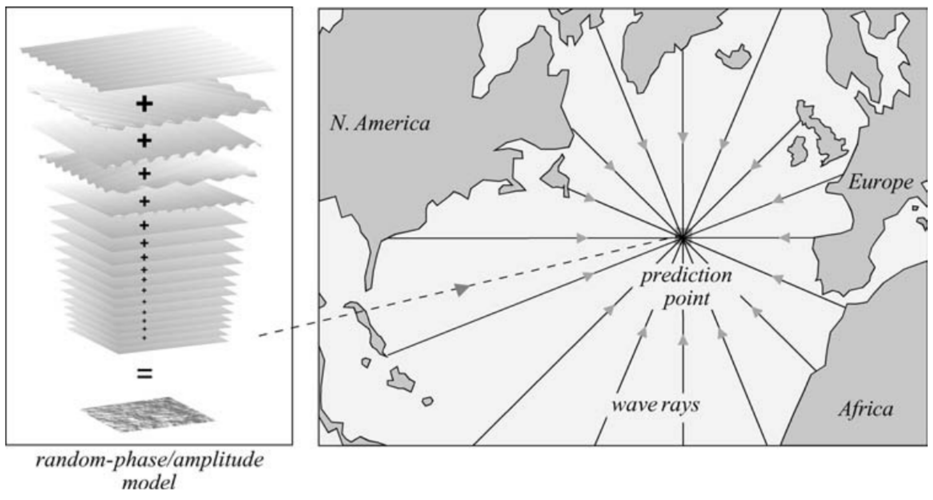


Figure 2.16: Following all wave components as they traverse across the ocean. A brief schematic outlining the process involved in predicting wave spectrum at a given point [Holthuijsen, 2007].

The phase-averaged model provides a robust and effective way to replicate wave transformation over a given geographic domain for a given time. Considering the minimal computational requirement (in relative terms) and efficient prediction of wave transformation parameters, the phase averaged models provides sufficient detail for limited computational effort.

Advantages

- Geographic resolution is flexible for implicit time discretizations. This makes the model design scalable in resolution as required
- Computational effort is lower
- No special requirement in terms of grid points for representing short wavelengths in the model

Shortcomings

- Parameterized source and sink terms (right hand side of the equation 2.8)

- Based on the linear wave theory which assumes linear waves while the waves within the surf-zone are almost always non-linear in nature, thus requiring parameterization of such non-linear processes.
- Output variables are extracted based on the associations and correlations between the wave spectrum and the variable of interest

2.5.2 Phase Resolving Models

Phase resolving models resolve the free surface [$\zeta(x, y, t)$] at the discretized points, unlike the phase averaged models which only compute the statistics based on the wave-action/wave-energy balance. The most widely used unsteady flow models have been based on the Boussinesq equation which are depth averaged version of the original 3-D wave equations. This limits the applicability of Boussinesq type models to shallow water regimes (Depth $\not>$ Wave Length). However, the developments in SWASH no longer limit (to a certain extent) the usage of phase resolving models to the shallow water criterion (See Zijlema et al. [2011] for details).

Simplified version of the Navier-Stokes equations yields the nonlinear shallow water equations. This simplified form leads to a faster computation at the cost of loss of information. This set of equations only yields the depth averaged velocity. The phase resolving model thus provides a reasonable improvement over the phase averaged description of the flow while still limiting the computational time.

Advantages

- Resolves the phase of the waves explicitly at a finer resolution in comparison to the phase averaged model
- Can simulate complex near-shore processes
- Relatively efficient for large computational domains and long term wave modeling applications

Shortcomings

- Wave-breaking is not explicitly resolved, but modeled using a roller analogy or propagating bore approach
- Not all models perform in full 3D mode. Most models work in depth-averaged modes.

2.5.3 Computational Fluid Dynamics Models

Computational Fluid Dynamics (CFD) can provide a complete description of fluid flow within the mathematical framework of the Navier-Stokes (NS) equations. As opposed to the previous two models, CFD is capable of resolving all the flow scales in the computational domain. However, for practical reasons, in most cases the information about the extremely small (dissipation/Kolmogorov) length scales is not required. Consequently, the generation and dissipation mechanisms can be modeled using the Reynolds-Averaged Navier-Stokes (RANS) equations with suitable closure approaches for the fluctuating turbulent component . These set of equation along with the mass conservation (and possibly energy, scalars etc) constitute a CFD modeling approach.

In this case, the interaction of air-water (interface) is important thus demanding a multiphase fluid flow model with simultaneous interface capture capabilities. In order to model the multiphase flow system, the flow field should satisfy the incompressible continuity equation 2.9 and the RANS equation 2.10.

$$\nabla\cdot\mathbf{u} = 0 \tag{2.9}$$

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla\cdot\rho\mathbf{u}\mathbf{u}^T = -\nabla p + \rho\mathbf{g} + \nabla\cdot[\mu\nabla\mathbf{u} + \rho\boldsymbol{\o}] + \sigma_T\kappa_\gamma\nabla_\gamma \tag{2.10}$$

where $\mathbf{u} = (u, v, w)$ is the velocity field in Cartesian coordinates, ρ is the density, p is the total pressure, $\mathbf{g}$ is the gravitational acceleration, μ is the dynamic fluid viscosity, and $\boldsymbol{\o} = -\overline{\mathbf{u}'\mathbf{u}'^{\mathbf{T}}}$ is the specific Reynolds stress tensor. Here, superscript T means the transpose of a vector. The choice of turbulence closure will be presented in the section 2.5.3. Numerical treatment of RANS closure will be discussed in section 2.5.3. The last term $(\sigma_T\kappa_\gamma\nabla_\gamma)$ in Equation 2.10 represents the surface tension interaction between the two fluids under consideration here (Air and Water).

As detailed in Rusche [2002], the use of total pressure introduces excess pressure as seen in Equation 2.11. Here $x = (x, y, z)$ is the coordinate vector.

$$p^* = p - \rho\mathbf{g}\cdot\mathbf{x} \tag{2.11}$$

Multiphase interface resolution follows a volume of fluid (VOF) approach which was introduced by Hirt and Nichols [1981], which follows an indicator (scalar quantity) passive advection to keep track of the interface between two/more fluids. A detailed description can be found in Hirt and Nichols [1981] and also in Jacobsen [2011]. The original implementation by Jacobsen [2011] used an interface tracking method which was based on the classical VOF method by Rusche [2002]. However, this thesis integrates the work by Røenby et al. [2016] as detailed in Section 2.5.3.

Wave generation and absorption

In order to impose wave boundary conditions[7], the wave generating capabilities within the `waves2Foam` toolbox developed by Jacobsen et al. [2012] were used. The reasoning behind using this approach for wave generation and absorption has been detailed in the following section. There are two basic (most important) requirements in order to successfully model wave flows:

- Appropriate wave generation thus imposing the right velocity, pressure, and phase volume fraction boundary condition (which are a transient set of boundary conditions)
- A way to handle the (if any present) reflected waves from the internal domain at the wave boundary generating location

In order to achieve the goal of wave generation and absorption, two fundamentally different approaches can be used as detailed in Miquel et al. [2018].

- **Passive Absorption :** Involves Sponge layers at the boundaries to absorb the reflected wave energy.
- **Active Absorption :** Generation of a wave opposite to that of the reflected wave, in order to cancel the reflected wave thus absorb wave energy.

[7] This part of the text is based on the work by Jacobsen [2011] with some minor additions to the current implementation of `waveFlow`

The details of the *passive* wave generation and absorption technique will be discussed in the following section. A potential flow solution of the chosen wave theory is first computed for arbitrary depth and values for $\mathbf{u_p}$, η_p, and $\frac{\partial p_p}{\partial n}$ are prescribed. Here the subscript p refers to the potential flow solution. Assuming that the length scale of the surface elevation is larger than the discretized boundary face (see Figure 2.17), the boundary face is intersected exactly at two points by the surface elevation (η_p).

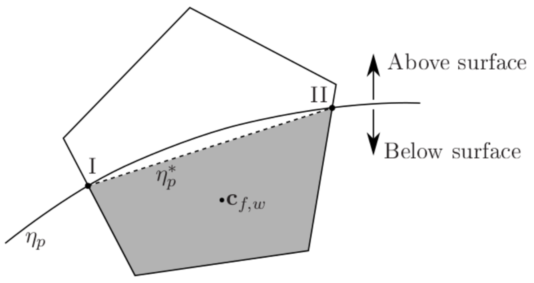


Figure 2.17: Sketch of a boundary face intersected by the surface, η_p , giving rise to a subdivision into a wet and a dry part. The corresponding wet centre, $mathbf{c}_{f,w}$, is of particular interest. Note that the boundary face can be an arbitrarily shaped convex polygon. Reproduced from [Jacobsen, 2011].

In order to prescribe an appropriate value for the phase volume fraction (γ), the intersection points are calculated and the ratio of the wet part of the boundary face (shaded part in figure 2.17) $\mathbf{A}_{f,w}$ to the total area of the boundary face $\mathbf{A}_f$ is used to set the alpha value at the boundary. This avoids oscillatory behavior at the boundary [Jacobsen et al., 2012]. The value for $\mathbf{u_p}$ and $\frac{\partial p_p}{\partial n}$ are evaluated for the wet part of the boundary face and the entire boundary face is assigned the same value. The boundary condition for the dry part of the boundary face has a boundary condition as detailed in Equation 2.12.

$$\mathbf{n} \cdot \nabla p_p = 0, \mathbf{u} = 0, \gamma = 0 \tag{2.12}$$

The second challenge in wave related flows is handling the reflected waves from within the domain. There exist a variety of ways in which the wave absorbing boundary condition/region can be handled (Adapted from [Jacobsen, 2011]).

- Apply the Sommerfeld boundary condition
- Sponge layer/Relaxation zone with appropriate damping functions
- Sponge layers/Relaxation zone with appropriate blending functions

The first of the above mentioned approaches only works as long as the long wave assumption holds. As a result, applying the Sommerfeld boundary condition is not a viable option since for a given wave climate, there is a mix of short (wind) waves along with some long waves. The use of sponge layers with damping functions requires the calibration of damping and the width of the sponge layer. Consequently, the use of blending function along with the relaxation zone proves to be a better fit for the type of wave modeling applications used in this study. The use of blending functions within the relaxation zone can modify an incoming wave field to a target function. The allows the use of this type of formulation at the inlet boundary to handle reflected waves along with its use at the outlet boundary. The relaxation zones work as a blend between a target solution and the computed solution. An example of the relaxation zone setup can be seen in Figure 2.18. The details of the implementation can be seen in Jacobsen [2011].

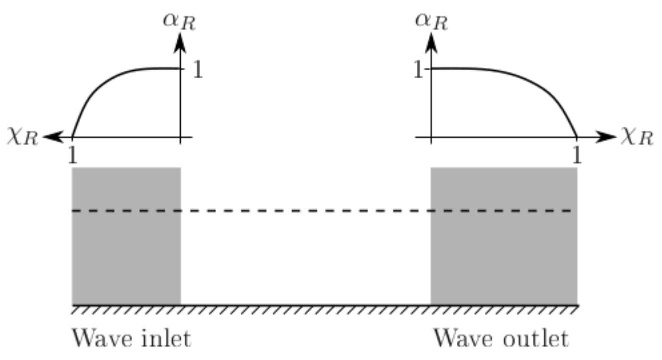


Figure 2.18: Sketch of the variation of the blending function, α_R, in both the inlet and outlet relaxation zones.

As discussed in the preceding section, there are multiple methodologies available within the CFD framework to generate and absorb waves in a numerical wave tank. However, each of these techniques has limitations on how it handles reflection. Miquel et al. [2018] showed that the two distinct methods of wave generation and absorption have different behavior based on the type of wave present in the numerical wave tank. To summarize this section, the following conclusions can be drawn [Miquel et al., 2018]:

- The relaxation zone technique provides better wave generation and absorption techniques for short and steep waves. However, this comes at the expense of higher computational cost in comparison to other techniques mentioned above.
- The active wave absorption technique developed by Higuera et al. [2013] within the `OpenFOAM` framework provides an efficient result in comparison to the relaxation zone technique as introduced by Jacobsen et al. [2012] for long waves.
- For analysis of breaking wave kinematics, the relaxation zone technique seems to be more suitable Miquel et al. [2018].

Given that the focus of this thesis is to first analyze the breaking wave hydrodynamics and extend the simulation for short (wind) waves, the use of the relaxation zone technique for wave generation and absorption will be used based on the work done by Jacobsen et al. [2012].

Turbulence modelling

As discussed in Section 2.3, turbulence is a result of the non-linear growth of instabilities which give rise to chaotic motion, eventually transforming to heat where most part of the energy is dissipated. In most numerical models using a RANS approach, this generation, transport, and dissipation of turbulent kinetic energy is modeled by means of convection-diffusion equations of scalars. A simple equation describing this convection-diffusion process can be seen in Equation 2.13.

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\overline{V}\phi) = S_\phi - D_\phi + \nabla \cdot (\kappa \nabla \phi) \tag{2.13}$$

where ϕ is the transported scalar quantity, S_ϕ is the generation of the scalar quantity, D_ϕ is the dissipation of the scalar quantity , and κ is the diffusivity constant. Each turbulence model has such a transport equation detailing the motion of the specified scalar. The scalar in this case can be the turbulent kinetic energy (k) or the specific dissipation rate (ω) in the case of the standard $k-\omega$ closure. The turbulence closure consists of two such convection-diffusion equations for k and $\omega = k/\epsilon$ respectively [Speziale et al., 1990]. The eddy viscosity is used to model the Reynolds stresses through the relationship

$$\boldsymbol{\o} = 2\nu_t \mathbf{S} - \frac{2}{3}k\mathbf{I} \tag{2.14}$$

where $\mathbf{S}$ is the mean rate of strain of the flow. Thus the molecular viscosity is added to the eddy viscosity in this approach. As a result, the diffusive transport term in the scalar convection-diffusion equation can be simply expressed as follows for k and ω respectively

$$\nabla \cdot [(\mu + \rho\sigma^* \nu_t)\nabla k] \tag{2.15}$$

$$\nabla \cdot [(\mu + \sigma_\omega \frac{k}{\omega})\nabla \omega] \tag{2.16}$$

Three distinct flow regimes can be observed in free surface wave propagation as seen in Figure 2.19. Region of potential flow is characterized with finite strain, consequently, bounded turbulence kinetic energy (TKE) is expected in these regions. However, the numerical investigations using a RANS approach have shown marked tendency to overpredict the turbulence levels within this potential flow region [Larsen and Fuhrman, 2018]. Higher turbulent kinetic energy in regions of potential flow regime introduce erroneously higher turbulent viscosity. As per the diffusion term in Equation 2.15. the eddy viscosity is added to the molecular viscosity. As a result, any unbounded growth in TKE results in higher eddy viscosity through the relationship

$$\nu_t = \frac{k}{\tilde{\omega}} \tag{2.17}$$

this effectively results in a thicker fluid (higher viscosity) and thus damping of the propagating wave. Consequently, controlling the unbounded growth in TKE values in regions of potential flow is important.

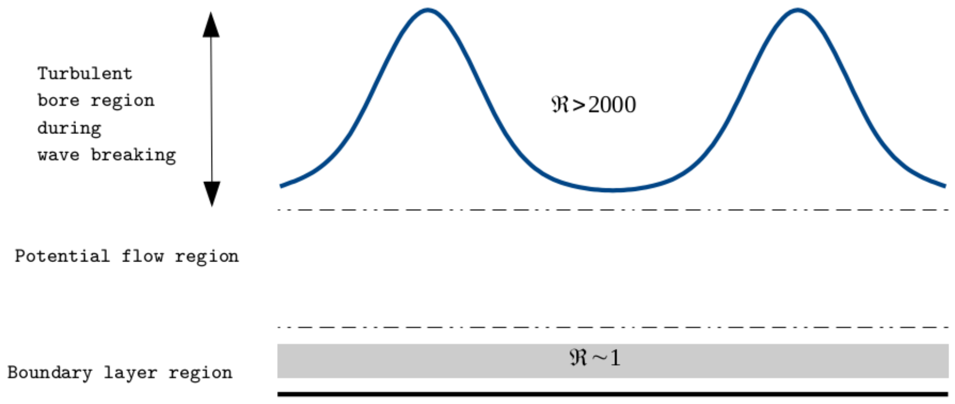


Figure 2.19: Schematic depicting the three distinct flow regimes in wave propagation. Here $\Re$ indicates the corresponding Reynolds number regime.

This problem was initially investigated by Mayer and Madsen [2000], where they showed analytically the unstable behavior of the standard $k - \omega$ based closure when applied in a region of potential flow. This lead to an exponential growth in TKE and thus the eddy viscosity. First attempts were made by Madsen et al. [1997] to resolve this issue by including the mean rotation rather than the strain rate. Further attempts to solve this problem were made by Devolder et al. [2018], however, this did not solve the problem completely. The overproduction of TKE was tackled by the investigation done by Larsen and Fuhrman [2018] where they showed that all two equation based closure models suffered from this issue and also provided the stabilization approach to avoid unbounded growth of TKE before wave breaking in the potential flow region.

In order to limit the production of TKE in the potential flow region, Larsen and Fuhrman [2018] proposed the two stress limiting coefficients viz., λ_1 and λ_2. The modified closure proposed by Wilcox [2006] introduced a stress limiter (λ_1) thus limiting the eddy viscosity in the regions where turbulence production exceeded the dissipation. Larsen and Fuhrman [2018] proposed a generalized $\tilde{\tilde{\omega}}$ as follows

$$\tilde{\tilde{\omega}} = max[\omega, \lambda_1 \sqrt{\frac{p_0 - p_b}{\beta^*}}] \tag{2.18}$$

in addition to this term, a modification to the $\tilde{\omega}$ in Equation 2.17 was also proposed

$$\tilde{\omega} = max[\tilde{\tilde{\omega}}, \lambda_2 \frac{\beta}{\beta^*} \frac{p_0}{p_\Omega} \omega] \tag{2.19}$$

where $\lambda_2 << 1$ is the additional stress limited which defines the threshold of p_Ω / p_0. This identifies the potential flow region. Consequently, the turbulence closure is stable as long as the condition in Equation 2.20 is true. The value for $\lambda_1 = 7/8 = 0.875$ as proposed by Wilcox [2006], while the value for λ_2 should be naturally small but at the same time also large enough for practical applications. This is because even in purely potential flow regions where $p_\Omega = 0$, the growth rate of TKE is given as $\Gamma_\infty = -\beta^* \omega_\infty$. As a result, some production is always expected even in purely potential flow regions. Hence, a value for $\lambda_2 < 0.2$[8] can be used for practical applications [Larsen and Fuhrman, 2018].

$$\frac{p_\Omega}{p_0} \leq \lambda_2 \tag{2.20}$$

In order to specify boundary condition values for the inlet phase, the following formulation replicated from Larsen and Fuhrman [2018] has been used. The TKE inlet specification is computed using the period and depth averaged production as mentioned in Equation 2.21.

$$\langle\langle p_0 \rangle\rangle = \frac{k_w^2 H^2 \sigma_w^2}{2tanh(k_w h)} \tag{2.21}$$

where $\langle\langle p_0 \rangle\rangle$ depth and period averaged turbulent kinetic energy production, k_w is the wave number, H is the wave height, σ_w is the angular frequency, and h is the water depth. The value for specific dissipation rate is given by the expression

$$\omega_\infty = 2.71 \sqrt{\langle\langle p_0 \rangle\rangle} \tag{2.22}$$

[8]The upper limit for $\lambda_2 \sim 0.2$ is a practice limit and has not been mentioned in the investigation by Larsen and Fuhrman [2018]. This value stems from personal communication with Larsen and Fuhrman [2018].

The details of the turbulence closure and the definition for the closure coefficients can be found in Wilcox [2006] and Larsen and Fuhrman [2018].

The results from Larsen and Fuhrman [2018] show improved performance of the wave breaking, undertow, and TKE profiles within the surf zone. However, getting the wave kinematics along with the right undertow profiles seems to be an unsolved problem still. Larsen and Fuhrman [2018] also acknowledge the discrepancy in the undertow profile prediction by the stabilized closure. However, given that the overproduction of turbulence is limited in the new stabilized closure, these new set of closures will be used in the thesis work.

Interface advection for two-phase flows

Multiphase flows within CFD provides a wide range of methodologies to handle the multiple fluid phases. One of the most popular methods to handle interface between two or more fluids is the volume-of-fluid (VOF) approach. In this method, an initially specified volume fraction is redistributed within the computational domain. The VOF technique can be further be divided into two categories:

- **Geometric method**: Explicit reconstruction of the interface from the volume fraction data. Generally more expensive and difficult to implement.
- **Algebraic method**: No such reconstruction employed. Generally less expensive and easier to implement.

As detailed in Larsen et al. [2018], the `interFoam` solver suffers from diffusive interface capture. This can have negative impact on the wave overtopping measured as a result of excessive flux registered over the overtopping faces. This study also details the effect of using low Courant number for long term wave propagation as it results in oscillatory behavior around the interface. Considering these issues in the numerical treatment of the interface, an alternative approach will be used to capture the free surface in this study. The interface advection method used is based on the work done by Røenby et al. [2016][9] and it retains the accuracy of the geometric schemes keeping the corresponding operations to a minimum. The details are as discussed in the sub-section below. The `isoAdvection` approach is divided into three fundamental steps as listed below:

VOF formulation

For a given computational domain $\mathcal{D} \in \mathbb{R}^3$ which contains a surface $\mathcal{S}$ embedded in it. Assuming two immiscible fluids $\mathcal{A}$ and $\mathcal{B}$, the surface $\mathcal{S}$ follows $\mathcal{A} \cap \mathcal{B} = \mathcal{S}$ and $\mathcal{A} \cup \mathcal{B} = \mathcal{D}$. This surface can also be represented as density field such that fluid $\mathcal{A}$ has density ρ_A while fluid $\mathcal{B}$ has density ρ_B. Naturally, the interface is the discontinuity since the two fluids are immiscible and only one fluid can exists at a given location, consequently demanding only one density value at a given spatial location. The evolution of the interface is thus given as an integral form of the continuity equation,

$$\frac{d}{dt}\int_{\mathcal{V}} \rho(\boldsymbol{x},t)dV = -\int_{\partial\mathcal{V}} \rho(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t)\cdot d\boldsymbol{S}, \tag{2.23}$$

here the equation simply prescribes a mass conservation formulation such that the rate of change within an arbitrary volume $\mathcal{V} \in \mathcal{D}$ is equal to the flux of the mass through its boundary $\partial\mathcal{V}$. In this case of purely advecting volume-fraction problem,

[9]This part of the text is based on the work done by Røenby et al. [2016]

the velocity field is pre-determined, hence, the different densities can be excluded from the description by means of an indicator field as described below,

$$H(\boldsymbol{x},t) \equiv \frac{\rho(\boldsymbol{x},t)-\rho_B}{\rho_A-\rho_B}, \tag{2.24}$$

here the indicator field $H = 1$ for all $\mathbf{x} \in \mathcal{A}(t)$, and $H = 0$ for $\mathbf{x} \in \mathcal{B}(t)$. In order integrate over the volume, some mesh definitions need to be placed. For a given computational grid as shown in Figure 2.20, the shared boundary between two cells is called as an *internal face*, while the shared faces between the cell and the boundary of the domain are called as *boundary faces*.

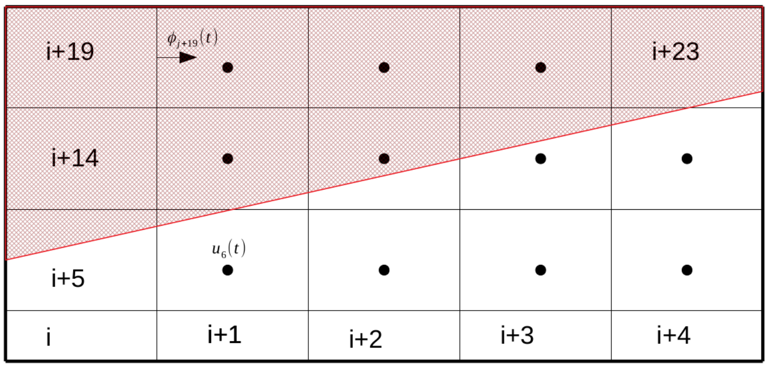


Figure 2.20: Schematic of a simplistic computational grid depicting the boundary of the computational domain (thick black line) while the red line indicates a fictitious fluid interface. The hatched red region represents one of the fluid with density (ρ_A) while the empty white region represents fluid with density (ρ_B)

Using the prescribed mesh definition (also see Rønenby et al. [2016]), the volume integration can be expressed in terms of the cell (say i),

$$\frac{d}{dt}\int_{\mathcal{C}_i} H(\boldsymbol{x},t)dV = -\sum_{j\in\mathcal{B}_i} s_{ij}\int_{\mathcal{F}_j} H(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t)\cdot d\boldsymbol{S}, \tag{2.25}$$

using the indicator field and the volume average, the volume fraction can be defined as,

$$\alpha_i(t) \equiv \frac{1}{V_i}\int_{\mathcal{C}_i} H(\boldsymbol{x},t)dV, \tag{2.26}$$

here the subscript i denotes the cell index that will be used to identify the cell within the computational domain. Now integrating Equation 2.25 from time t to $t+\Delta t$ after substitution of the volume fraction definition yields,

$$\alpha_i(t+\Delta t) = \alpha_i(t) - \frac{1}{V_i}\sum_{j\in\mathcal{B}_i} s_{ij}\int_t^{t+\Delta t}\int_{\mathcal{F}_j} H(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t)\cdot d\boldsymbol{S}d\tau, \tag{2.27}$$

this form above represents the exact form (without any numerical approximations) of the interface advection problem which needs to be solved. The time integration on the right hand side of the equation solves for α_i and thus the surface $\mathcal{S}$ in time. Using additional simplification for the time integral (see Rønenby et al. [2016]), the fundamental equation for volume fraction can be represented as,

$$\alpha_i(t+\Delta t) = \alpha_i - \frac{1}{V_i}\sum_{j\in\mathcal{B}_i} s_{ij}\Delta V_j(t,\Delta t) \tag{2.28}$$

using the above formulation for the volume fraction and the cell averaged velocity ($\mathbf{u}_i(t)$) along with the face flux ($\phi_j(t)$), `isoAdvection` represents one of the most effective ways to compute the transport of the fluid (α_i) across the domain in the time interval $[t, \Delta t]$.

Interface reconstruction step

The distribution of the volume fraction using the approach as described above can result in non-singular configurations within a given computational cell. For example, a volume fraction of 0.5 can be represented in multiple ways as seen in Figure 2.21.

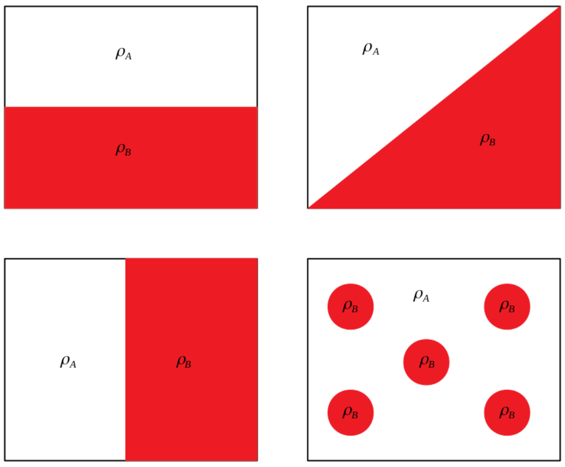


Figure 2.21: Fluid representations within a computational cell for the same volume fraction of $\alpha_i = 0.5$ with different configurations.

Considering this issue, a subgrid model which can handle these interfaces is required to differentiate the various configurations based on information available at time t within the given cell. An implicit assumption which is quite important in the application for wave models is that the *local radius of curvature is larger than the cell size* Røenby et al. [2016]. Consequently, resolving individual blobs of fluid smaller than or equal to the grid cell size become infeasible using this approach. However, in most cases, this level of detail is not required.

Using an iso-surface for a given volume fraction (say $\alpha_i = 0.5$) can provide a good estimate of the local fluid distribution as along as the assumption mentioned above is satisfied. However, this demands a logical choice for the volume fraction value. To resolve this issue, Røenby et al. [2016] propose a method to find an optimal volume fraction value which represents the local distribution of the volume fraction within the cell. It is important to note that this results in a curve which is not continuous as would be in curve connected by the same volume fraction values.

Advection step

In order to advect the volume fraction values, the integrand in Equation 2.27, demands the velocity field at time $t + \Delta t$. However, in most segregated solvers the velocity information is only available up until time t. Assuming that the velocity is constant over the time step[10] and the face flux is defined as,

$$\boldsymbol{u}(\boldsymbol{x},t)\cdot d\boldsymbol{S} \approx \frac{\phi_j(t)}{|\boldsymbol{S}_j|}d\boldsymbol{S} \qquad \forall \boldsymbol{x}\in\mathcal{F}_j \qquad \text{and} \tag{2.29}$$

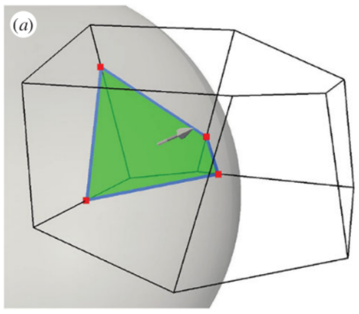
using this information, the following can be formulated (for details see Rœnby et al. [2016]),

$$\Delta V_j(t,\Delta t) \approx \frac{\phi_j(t)}{|\boldsymbol{S}_j|}\int_t^{t+\Delta t} A_j(\tau)d\tau. \tag{2.30}$$

A critical assumption is that the velocity field is constant in both time and space. Equation 2.30 is exact, which means if the cell becomes sufficiently small compared with the temporal variations in the velocity field, the error committed in the above formulation is negligible Rœnby et al. [2016]. The central challenge in most VOF methods as discussed by Rœnby et al. [2016] is to reconstruct the time evolution within a time step of the submerged are of the face followed by the integration of this area in time. The time scale on which $A_j(\tau)$ changes is dictated by:

- Orientation of the faces and the fluid interface
- Direction of motion of the interface
- Shape of the specific polygonal faces.

As discussed in the previous sections, the discontinuous nature of $A_j(\tau)$ makes $\Delta V_j(t,\Delta t)$ non-differentiable thus rendering the advection problem relatively difficult to solve. The `isoAdvection` algorithm calculates $A_j(\tau)$ for face j using the upwind cell of face j at time t for the first estimate. The motion of the isoface in a time step window of $[t,\Delta t]$ can be approximated by using the velocity data in the surrounding cells. Figure 2.22, shows one such example of advecting isoface at three different locations (taken from Rœnby et al. [2016]).



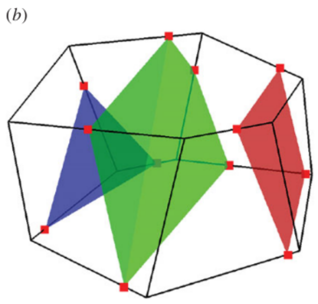


Figure 2.22: (a) A spherical surface cutting a polyhedral cell. Red dots are the edge cutting points. Blue lines are the face–interface intersection lines. Green patch is the isoface. (b) The isoface motion is estimated from surrounding velocity data and the isoface is propagated. Isoface at three different times within a time step are shown. [Rœnby et al., 2016]

This VOF technique promises shape preservation, volume conservation, boundedness, interface sharpness, and efficiency. Consequently, it will be used to carry out a comparative analysis for overtopping volumes against `waveFoam`.

[10]The assumption is based on the fact that velocity does not drastically change within a given time step. As a result, a small Courant number gives better description using this method. This is inline with most of the VOF approaches except for the development of oscillatory currents as discussed in Section 3.

2.5.4 Modeling large time and space domains

In most of the coastal studies an accurate description of the wave transformation from offshore conditions to onshore conditions is required. Replicating such a system by means of a numerical model requires the inclusion of the right offshore and onshore flow conditions. In addition to the large spatial domain, which is often times in the order of $\mathcal{O}(10^2)$ m[11], wave run-up and overtopping studies require large number of waves to derive statistical inferences and this would require a time domain replication of about 2 hours $\sim$ 7200 s. In most cases the Reynolds number for typical coastal flow applications can range from 10^5 to 10^9, thus rendering Direct Numerical Simulations computationally unfeasible. The requirement of large spatial and time domain can increase the computational time even with a RANS closure to unfeasible limits. Consequently, making any coastal investigations using fully non-linear CFD approach impractical.

Figure 2.23 shows the two distinct regions present in the cross shore region of a typical coast. Most of the offshore region is characterized by a 'potential flow' region where there is no turbulence present. Wave breaking and other deep water mechanisms can introduce non-linear growth of surface instabilities thus introducing turbulent motion across the depth of the fluid. Since there is relatively negligible turbulence present in the offshore regions, explicit resolution of the wave dynamics/kinematics in this region does not require a fully non-linear CFD model. It is only in the region close to wave breaking and deep inside the surf zone that the requirement of resolving wave breaking and associated physics becomes crucial. As a result, modeling large spatial and time domains can be optimized by coupling two different models. In this case, the offshore region can be numerically represented using a non-linear 'potential flow' model named OceanWave3D [Engsig-Karup et al., 2009] while the rest of the domain can be represented in the fully non-linear CFD model such as `OpenFOAM`. The details of the CFD model have been discussed in the the preceding section, the following section will discuss the computational details for the potential flow model.

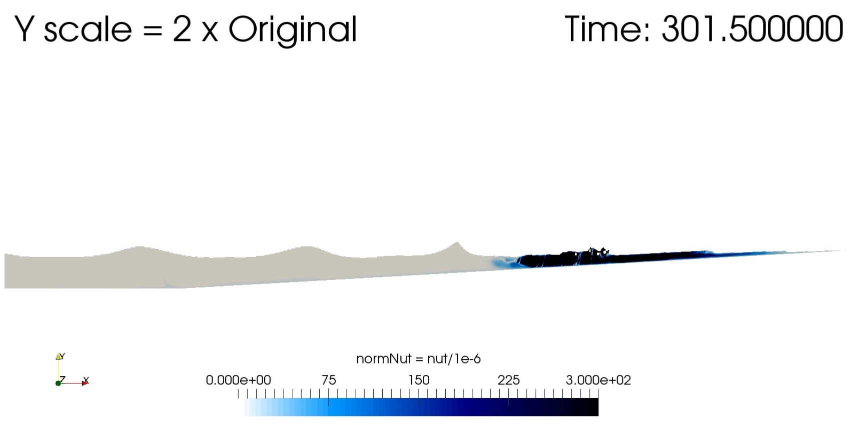


Figure 2.23: Spilling waves breaking over a slope conforming to the studies done by Ting and Kirby [1994]. The dark colored region represents normalized turbulent viscosity ν_t.

OceanWave3D is a flexible-order, finite-difference based numerical model which provides fully non-linear solution to potential flow problems for waves on a fluid of variable depth. The time varying physical domain is mapped to a time invariant boundary-fitted computational domain to obtain the numerical solutions for unsteady free-surface flows. Since wave overturning is not resolved due to the limitation of the single valued free surface, wave breaking is modeled by means of a

[11] at prototype scale

dissipation term in the system. In this context, it becomes appropriate to solve the Euler equations as opposed to solving the pure potential flow equations. Consequently, the current approach is transformed from a Laplace solver to a Poisson solver [Engsig-Karup et al., 2009].

Formulation of the problem

Using a Cartesian coordinate system[12] with the xy-plane located at the still water level and the z axis pointing upwards, the water depth can be defined using $h(\boldsymbol{x})$ with $\boldsymbol{x} = (x, y)$ the horizontal coordinate. Now assuming inviscid fluid and irrotational flow conditions, the fluid velocity $(\boldsymbol{u}, w) = (u, v, w) = (\nabla\phi, \partial_z\phi)$ can be used to define the gradient of the scalar velocity potential. Using this construction, the free surface evolution is governed by the kinematic and dynamic boundary conditions,

$$\partial_t \eta = -\nabla\eta \cdot \nabla\tilde{\phi} + \tilde{w}(1 + \nabla\eta \cdot \nabla\eta) \tag{2.31}$$

$$\partial_t \tilde{\eta} = -g\eta - \frac{1}{2}(\nabla\tilde{\phi} \cdot \nabla\tilde{\phi} + \tilde{w}^2(1 + \nabla\eta \cdot \nabla\eta)) \tag{2.32}$$

solving for $\tilde{w}$ and evolution of the system of equation requires the solution to the Laplace equations where $\tilde{\phi}$ and η are prescribed. Together with the kinematic boundary conditions, no flux across the solid walls boundary condition, the solution for the wave problems can be obtained (See Engsig-Karup et al. [2009] for details).

This approach ensures that the computational time is limited to feasible limits and large scale domains can be replicated in the numerical wave tank.

[12]This part of the text is mainly based on Engsig-Karup et al. [2009]

3 DEVELOPMENT OF WAVEFLOW

OUTLINE

This chapter provides a brief introduction to the process of combining the isoAdvection [Rønenby et al., 2016] capabilities along with the `waveFoam` [Jacobsen et al., 2012] solver. This will be followed up by a proof of concept test case which will aim at the comparison of `waveFlow` and `waveFoam`. The results section will capture the most important aspects such a comparison. In the concluding section, some motivation will be presented in order to use the new solver for further investigation of wave hydrodynamics.

3.1 WAVEFLOW DEVELOPMENT

The traditional waveFoam solver originally created by Jacobsen et al. [2012] was based on the generalized Volume Of Fluid (VOF) method to model multiphase wave flow. Since this solver was based on the interFoam solver in the `OpenFOAM` framework, it also suffered from the shortcomings of diffusive interface capture method. As outlined by Larsen et al. [2018], the interface capture in `interFoam` solver is smeared over several cells depending on the grid size used. In the original implementation, a compression velocity is used to limit this smearing. However, this leads to some undesired effects such as wiggles over the interface as seen in Figure 3.1. Although, this study brings to light the possibility of obtaining relatively good results, they show that this demands a relatively small Courant number and first order time marching (Euler type) to avoid unphysical wiggles in the solution. This directly translates to higher computational cost due to the lower Courant number criterion (also see Section 1.1) and higher time marching error due to diffusive nature of the Euler time discretization.

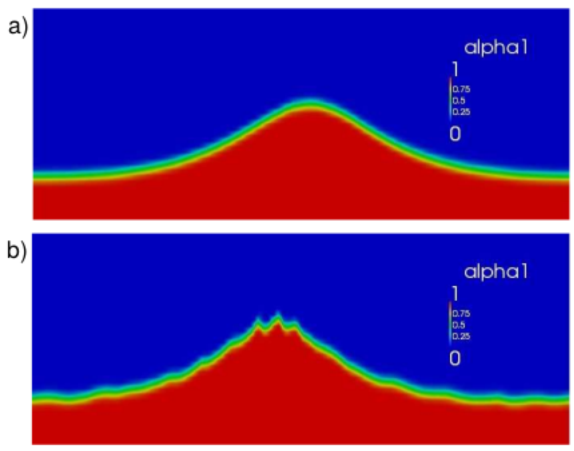


Figure 3.1: Snapshots at *a)* t = 5.5T and *b)* t = 16.25T, illustrating the appearance of small wiggles in the crest after sufficiently long propagation. Adapted from [Larsen et al., 2018].

As described in Section 2.5.3, the sharp interface captue technique can be used to avoid these numerical wiggles. This motivates the implementation of the isoAd-vection scheme developed by Rœnby et al. [2016]. Since the wave boundary conditions are already well established and validated in the waves2Foam library [Jacobsen et al., 2012] within OpenFOAM. Details of the integration of the two libraries have been detailed in Appendix A.

3.2 SENSITIVITY TEST SETUP

In order to assess waveFlow and its performance, a conceptual calibration test setup was though of and tested against waveFoam. The test case domain layout is as described in the Figure 3.2. Since wave surface elevation and overtopping discharge was assumed to be affected by the interface capture method used, these two parameters were assessed for this test case.

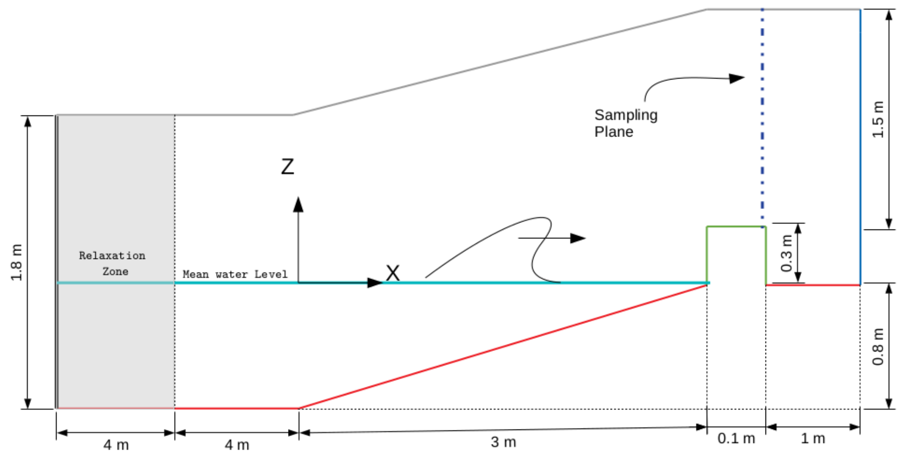


Figure 3.2: Schematic of the calibration model setup with a side view.

The incident wave conditions for the test case are as described in Table 3.1. Since the choice for choosing the wave types was free, the simplest wave types were chosen to minimize the complexity of the model. The sampling plane represents the location across which the overtopping discharge is assessed. The grid resolution used in both the solver simulations was 1 [cell] : 0.025 [m]. The stabilized turbulence closure models as developed by Larsen and Fuhrman [2018] have been used, specifically the $k-\omega$ model with the stability parameter of $\lambda_2 = 0.05$ has been used in this particular case. The overtopping function utility as discussed in section 3.1 has been used for the test case in order to compare the overtopping discharge across the sampling plane.

Table 3.1: Wave conditions

Solver	Wave Type	Wave Height [m]	Wave Period [s]
waveFoam	cnoidal	0.3	2.1
waveFlow	cnoidal	0.3	2.1

3.3 SENSITIVITY TEST RESULTS

In order to compare waveFlow, key parameters like surface elevation, overtopping discharge, cumulative overtopping volume, and fluid structure interaction behavior will be compared against waveFoam. Some of the previously listed parameters are qualitatively compared while some of them are quantitatively compared against each.

3.3.1 Surface Elevation

Surface elevation function utility within the `waves2Foam` library was used to measure the time series at specific locations within the computational domain. The wave gauge locations within the domain can be seen in Figure 3.3. A total of 100 wave gauge time series were sampled for the given locations. The time series was sampled every 0.001 *s* i.e., frequency of sampling = 1000 *Hz*.

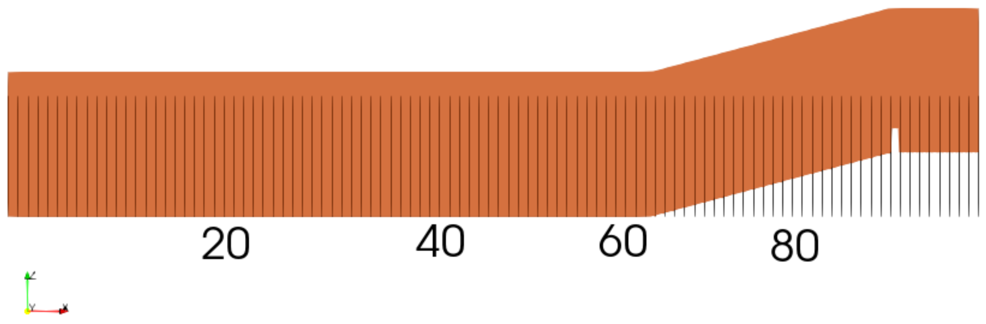


Figure 3.3: Depiction of wave gauge locations within the computational domain. The numbers below represent approximately the wave gauge number.

As seen in Figure 3.2, the first 4 *m* of the domain belong to the relaxation zone where all of the reflected spectrum is absorbed. It is clearly seen that there is no appreciable difference between the water surface elevation profiles within the relaxation zone. This is expected since, the relaxation zone will try to impose the incoming wave conditions where the free surface is explicitly specified by the boundary condition without significant wave transformation due to numerical and bottom surface changes. Both the solvers seem to behave identically within the relaxation zone.

However, comparison of the surface elevation time series outside the relaxation zones tells a different story altogether. As seen in Figure B.2, the waves are transformed due to the change in the bathymetry and also due to the reflection of the incident waves form the vertical structure and the sloping beach profile. A closeup view for a few waves at wave gauge number 60 can be seen in Figure 3.4. One wave gauge is located every $0.122\bar{2}$ m within the domain. As a result, wave gauge 60 is approximately located at x $\sim$ -0.66 m with reference to Figure 3.2. Although the figure 3.4 shows that the two solvers are able provide a general picture of the water surface elevation, the subtle differences between the wave shape and amplitude of the water surface elevation are to be noted. A video render of the differences between the flow field for interface capture have been published at the following hyper-link[1].

The wave transformation over the slope is clearly captured in Figure 3.5. This time series corresponds to wave gauge 82 which is located x $\sim$ +2 m in the computational domain. There seem to characteristic differences between the wave surface elevation amplitudes for the two wave solvers. It is also interesting to note that both the solvers almost identically follow each other with respect to the wave behavior.

[1]Link to the video: https://youtu.be/CHa57xGN89o

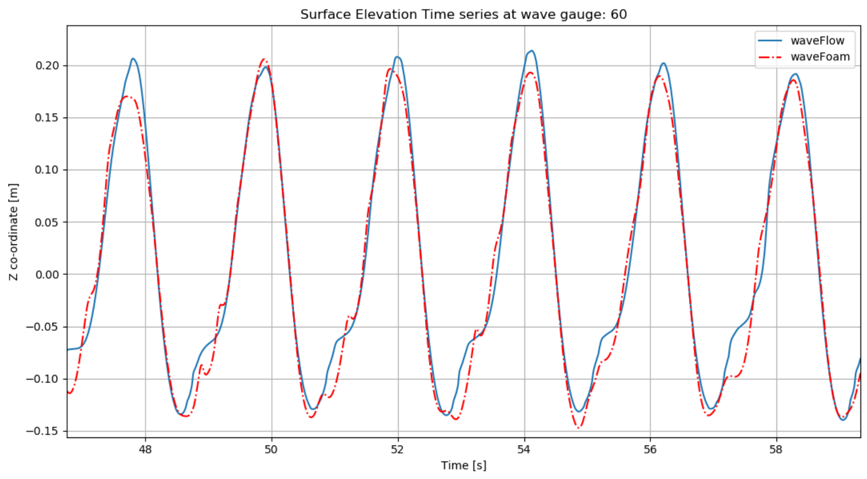


Figure 3.4: Close up view of the surface elevation comparison for wave gauge 60 which is located at x ∼ -0.66 m in the computational domain.

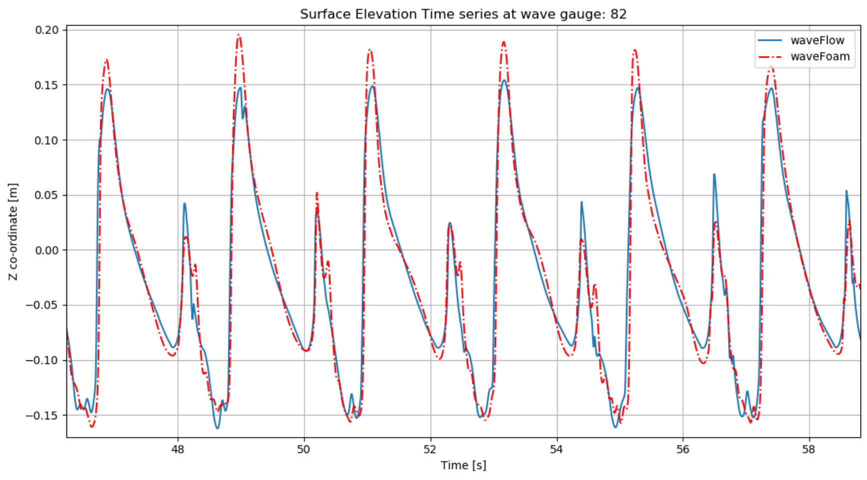
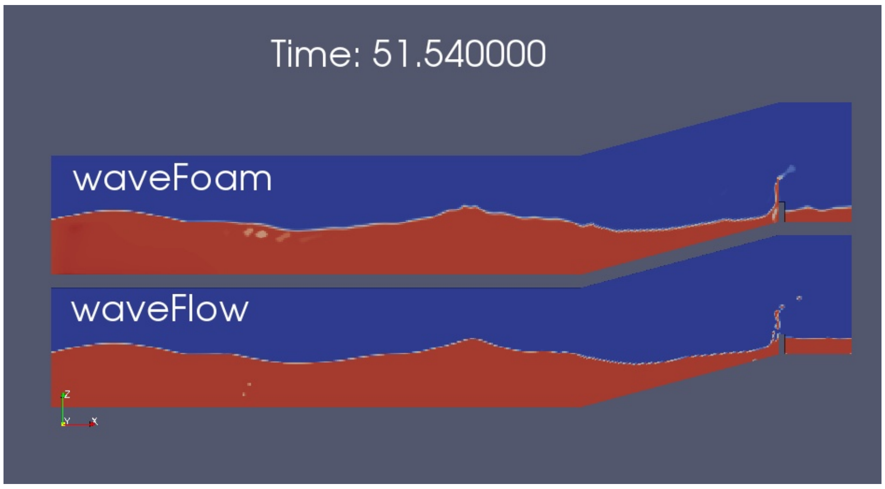


Figure 3.5: Close up view of the surface elevation comparison for wave gauge 82 which is located at x ∼ +2.0 m in the computational domain.

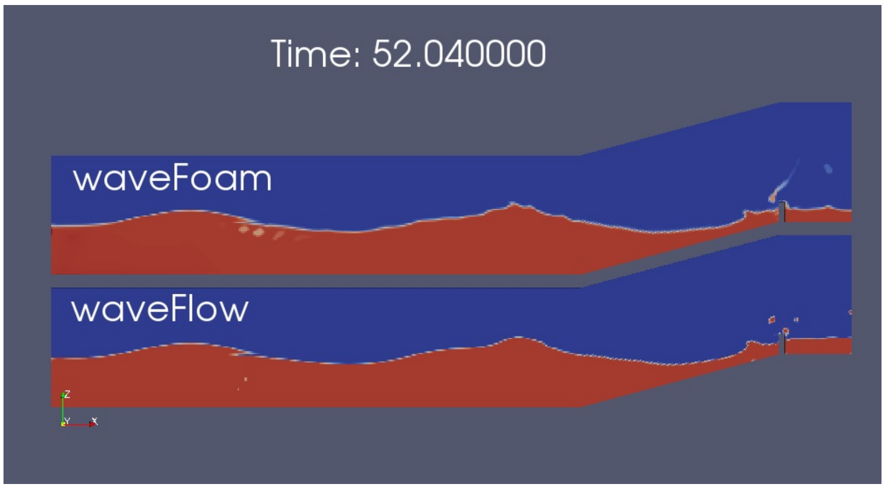
3.3.2 Overtopping behavior

Overtopping is sensitive to the approach which is used to resolve the air-water interface in the numerical model. Once the wave impacts the structure (See Figure 3.6), the horizontal momentum is generally transformed into splash of fluid going vertically upwards. The bulk of the fluid will be transformed into small but dispersed set of fluid parcels/packets. Given the limitation of the interface capture methods, obtaining this behavior in the numerical model can prove to be challenging. However, the use of `isoAdvection` can help overcome this difficulty to a certain extent. In order to compare the differences between the interface capture, a series of numerical model results are compared to observe the qualitative differences between the interface capture method. The behavior of the fluid after interacting with the vertical wall can be seen in the Figure 3.7.

Figure 3.6: Interaction of incoming waves with a solid boundary.



(a) T = 51.540 s



(b) T = 52.040 s

Figure 3.7: Comparison of the numerical results for the two solvers under investigations.

Two methods were used to address overtopping sampling for the given sensitivity test as listed below:

- Face flux measurement (F)
- Cell center velocity measurement ($U \cdot \alpha_w \sim U_{water}$)

The fundamental difference between the two parameters can be seen in the Figure 3.8 below. The face flux for each face depcits the conserved flux of fluid passing through the face which is also used to solve the momentum equation, while the cell center value is derived from a vector sum of the face flux values and stored at the cell center. Since the face flux is used to solve the equations, it gurantees mass conservation while the cell center velocity value does not always gurantee mass conservation. This demands the use of face flux value in order to sample the overtopping flux passing through the prescribed overtopping faces. In order to assess the differences between the two parameters, the analysis presented below was considered. Since the cell centered value does not gurantee mass conservation, the face flux approach will be considered in the following sections for overtopping measurement.

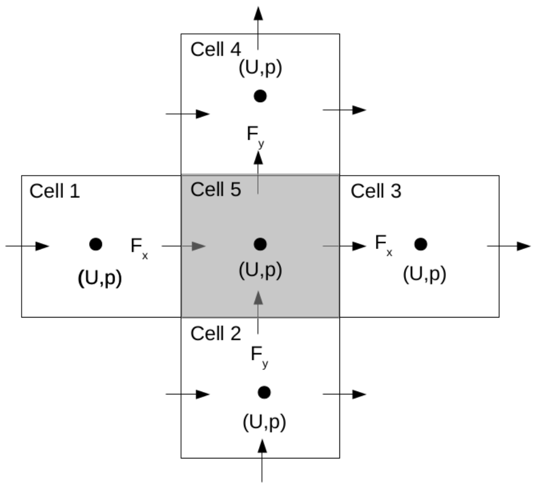


Figure 3.8: Computational grid depciting the face flux and the cell center velocity for a finite volume type discretization.

In addition to the visual inspection, the overtopping time series for one such wave cycle has been demonstrated in Figure 3.9. Complete time series and additional wave overtopping comparisons can be found in Appendix B. The rate of sampling for the overtopping function utility was varying since overtopping was logged every time step of the numerical model. In general the overtopping results are stored every $\sim$ 0.001 s of the simulation time. The discharge indicated in this figure is instantaneous discharge.

There are systematic differences between each of the approaches and solvers used to calculate the instantaneous overtopping for the test case. Some of the key observations can be listed below:

- The `isoAdvection` interface capture method is able to resolve the smaller parcels of fluid which can be easily seen in Figure 3.7.
- The interface capture by `isoAdvection` does not show any appreciable wiggles/oscillations which can be observed in the traditional VOF simulation. This can be seen in the Figure 3.7b.
- The calculation of overtopping when using the formulation $U_{water} = U \cdot \alpha_{water}$, does not result in the accurate computation of the instantaneous overtopping discharge. This is due to the difference between the face flux value and velocity value in the Finite Volume discretization method.
- Different behavior in terms of the amplitude and phase of the overtopping time series can be observed when comparing `waveFlow` and `waveFoam` solvers.

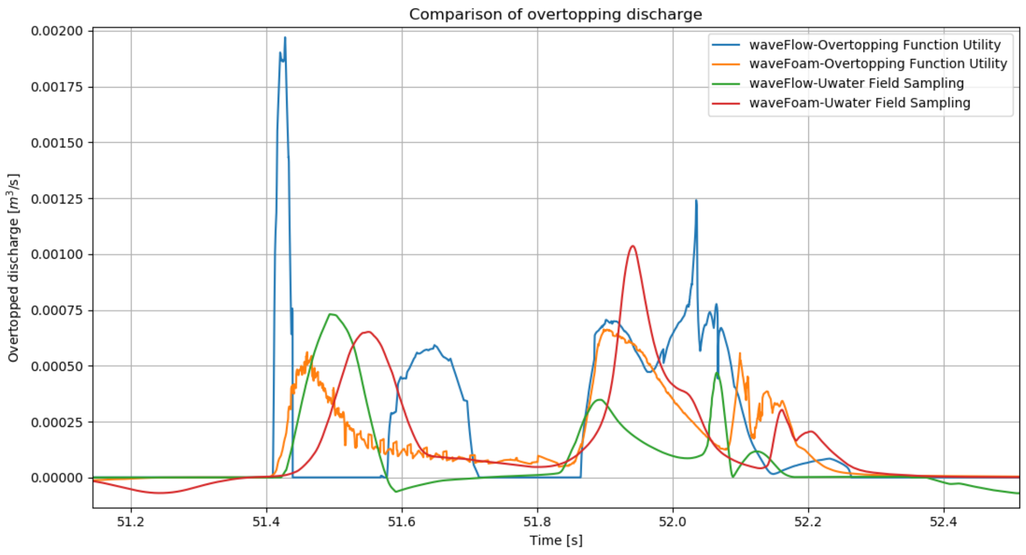


Figure 3.9: Overtopping function utility calculation for one wave cycle. The wave cycle corresponds to the wave at time T = 51 s.

Interestingly the water elevation comparison within the storage tank behind the vertical wall also displays very different behavior for the two solvers. Figure 3.10 shows that lower overtopping volume is predicted by the `waveFlow` solver in comparison to `waveFoam`. This can also be seen in Figure 3.11 that shows the cumulative overtopping time series.

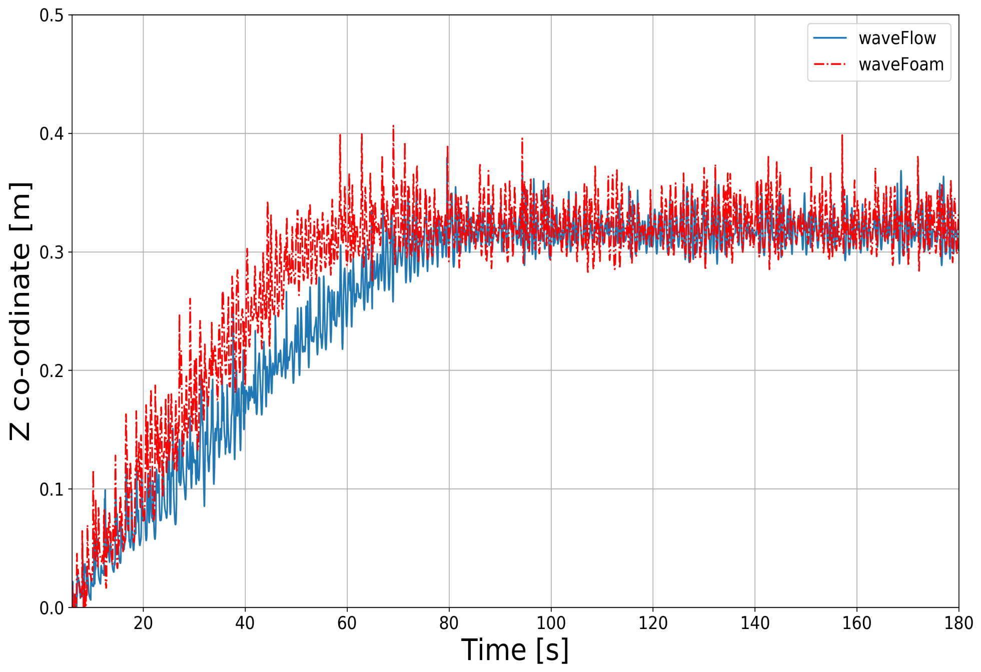


Figure 3.10: Water surface elevation inside the tank at location x = 11.61 m

The differences between the two solvers can be distinctly visualized by plotting the overtopped volume of water as a function of time. In order to arrive at the cumulative value of overtopped volume of water, equation 3.1 was used. The time series of the cumulative overtopping volume can be seen in Figure 3.11.

$$V = \dot{\mathbf{q}}\Delta t \tag{3.1}$$

where V = Instantaneous overtopped volume [m^3], $\dot{\mathbf{q}}$ is the instantaneous overtopping discharge [m^3/s], and Δt is the time step [s].

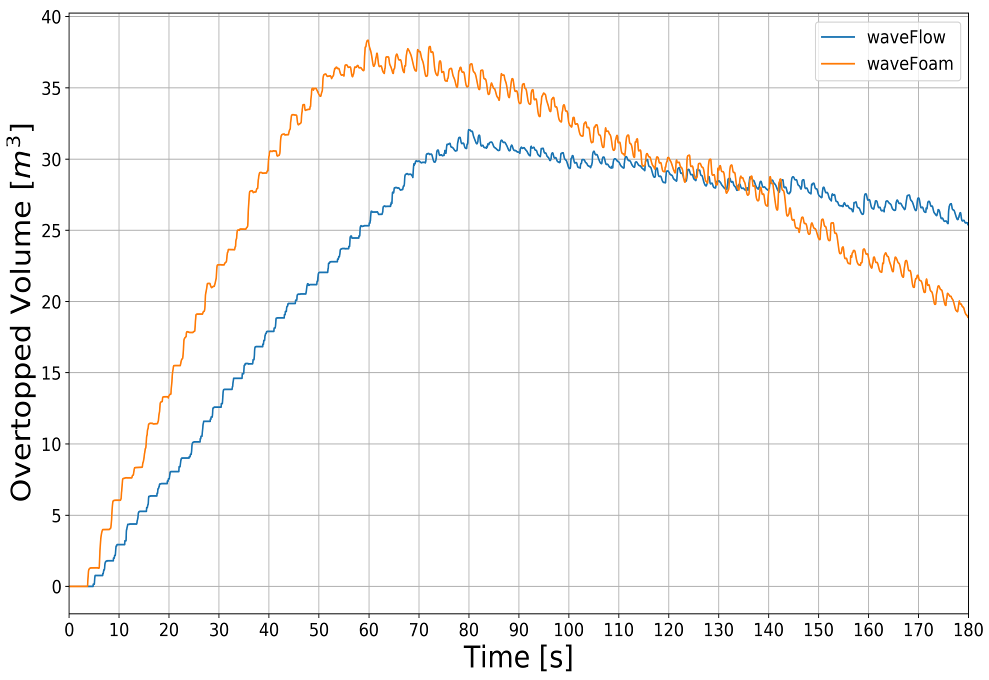


Figure 3.11: Cumulative overtopped volume time series for comparison of the two solvers. The cumulative volume has been computed using equation 3.1.

Despite the lack of any experimental comparison, this proof of concept test displays the crucial differences between the two solvers used to carry out wave simulations. It is interesting to note the overprediction using the `waveFoam` solver for identical model setup. This can be attributed to the diffsued nature of the interface. Based on this quick proof of concept test and the distinct overtopping behavior results, the new solver `waveFlow` was chosen to investigate the behavior for shallow foreshore environments.

4 NUMERICAL MODEL SETUP

OUTLINE

This chapter details the numerical model setup used in this particular investigation. A brief overview of each of the experimental setups and the corresponding numerical model settings will be discussed. Section 4.1 details the case setup followed by Larsen and Fuhrman [2018] for the Ting and Kirby [1994] experimental replication/validation. This will be followed by the Flanders Hydraulics experimental investigations.

4.1 REFERENCE CASE STUDY BY TING AND KIRBY [1994]

This experimental campaign focused on mean flow and turbulence under spilling and plunging breakers. The details of offshore wave conditions generated in the experimental studies by Ting and Kirby [1994] can be seen in table 4.1. A laser-doppler anemometer was used to extensively probe the flow field for velocity measurements. Periodic cnoidal waves were generated for at least 20 minutes before the sampling of data began. In order to minimize the error in ensemble averaged quantities, about 102 waves were averaged. Considering the quality of experimental data produced by Ting and Kirby [1994], benchmarking new developments in the numerical model can be based on this particular dataset.

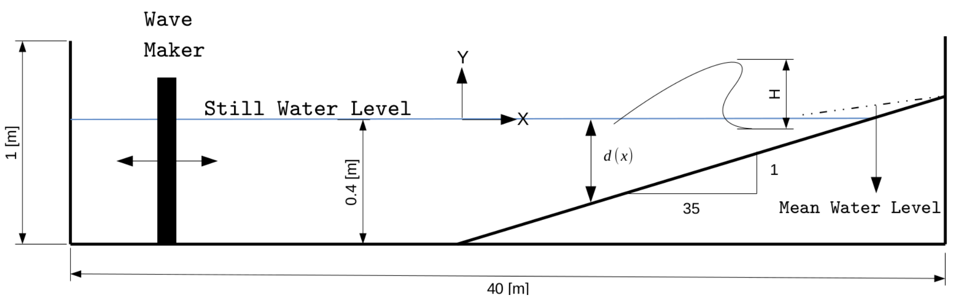


Figure 4.1: Schematic of the experimental setup with a side view. The experimental flume had a false bottom of plywood with 0.6 m width [Ting and Kirby, 1994].

Table 4.1: Wave conditions (subscripts o,h, and b denote deep water, horizontal region and breaking point) [Ting and Kirby, 1994]

Breaker Type	H_0 [m]	H_h [m]	**T** [s]	H_0/L_0 [-]	x_b [m]	d_b [m]
Spilling	0.127	0.125	2.0	0.02	6.4	0.196
Plunging	0.089	0.128	5.0	0.0023	7.795	0.156

Since new solvers in OpenFOAM have been used in the current thesis, a pre-calibration for qualitative understanding and baseline testing had to be consulted for understanding the deficiencies and improvements. This proof of concept test has been discussed in chapter 3. Based on the results and discussions from chapter 3, the new solver has been used for all further investigations in this thesis.

4.1.1 Numerical grid

The numerical flume is used to replicate the experimental setup as discussed in the previous section. The `blockMesh` utility in `OpenFOAM` was used to generate the numerical grid. The details of the grid and other setup can be found in Appendix C. The generated grid can be seen in Figure 4.2. The size of the first computational cell has been based on a similar study carried out by Larsen and Fuhrman [2018]. Since a RANS turbulence closure is employed, the wall boundary conditions for the turbulence parameters require a wall function. As discussed in section 2.5.3 and Larsen and Fuhrman [2018] and Brown et al. [2016], setting the first grid size equal to $7.5 \cdot 10^{-4} m$ ensures that the relationship in equation 4.1 holds [Larsen and Fuhrman, 2018].

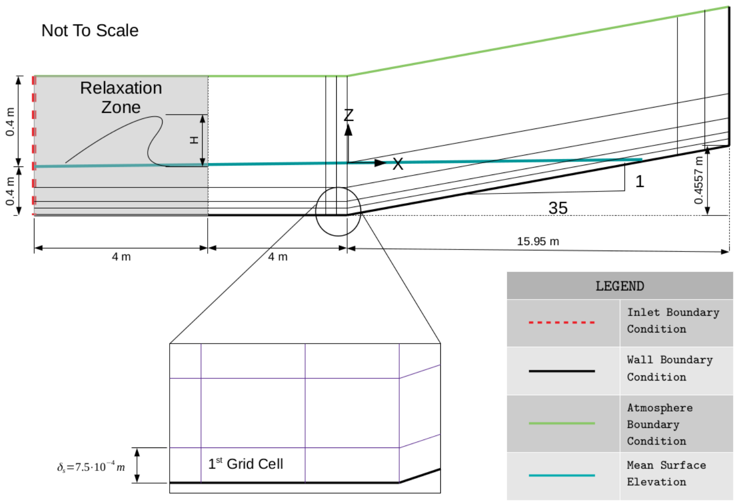


Figure 4.2: Computational grid generated using `blockMesh`. The schematic depicts a few grid features such as the use of inflation layers and the boundary conditions used in the model.

$$\Delta z^{+} = \Delta z U_f / \nu < 30 \tag{4.1}$$

In addition to eh specification of the z^{+} distance, the Nikuradse equivalent sand roughness has been set to $k_s = 10^{-4} m$, since the plywood false bottom was mentioned in the experimental setup [Ting and Kirby, 1994].

4.1.2 Turbulence initial conditions

The RANS $k - \omega$ turbulence closure with relevant stabilization [Larsen and Fuhrman, 2018] for wave applications has been used in this numerical model. Based on the discussions carried out by Brown et al. [2016] and Pengzhi and Philip [1998], initial conditions where $k = 0$ results in the transport equations (of turbulence quantities) being singular. Consequently, a finite amount of turbulence has to be initialized within the system. Using the formulations as discussed below, the initial conditions for the turbulence parameters have been initialized. The initial condition for ω is taken as $\omega = \omega_\infty = 2.71\sqrt{p_0}$, where p_0 can be estimated using equation 2.21. Using the values from table 4.1, the initial value for the production term is estimated to be $\langle\langle p_0 \rangle\rangle \sim 0.532$. Hence, the values for $\omega_\infty = 1.97\bar{7}$ and $k_0 = 0.1\omega_\infty \nu = 1.97\bar{7}e{-}07$.

The factor 0.1 in the estimation of k_0 is used to seperate the initial turbulence scales from the mean flow. The stabilized $k - \omega$ turbulence closure with the values for $\lambda_1 = 0.875$ and $\lambda_2 = 0.05$ is observed to perform adequately barring the overestimation/underestimation of k and ω inside the surf zone for certain parameters of λ_1 and λ_2 as detailed in Larsen and Fuhrman [2018]. Consequently, for all of the numerical studies in the following section, the values for $\lambda_1 = 0.875$ and $\lambda_2 = 0.1$ will be used.

4.1.3 Runtime data sampling

In order have a consistent comparison for the undertow profiles and surface elevation profiles, run-time sampling of the internal values was carried out. In order to do this, about 100 wave gauges were placed within the domain as seen in Figure 4.3. In addition to the wave gauges, several vertical sampling lines were introduced to measure the undertow profiles, turbulence levels, and velocity profiles as seen in Table 4.2.

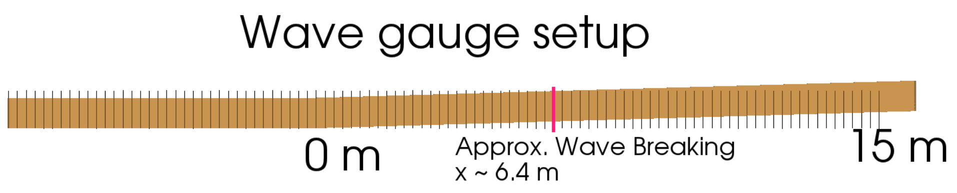


Figure 4.3: Wave gauge setup within the computational domain. The vertical black lines represents wave gauges, while the red line shows the approximate location of wave breaking for the spilling breaker case.

Table 4.2: Location of the vertical sampling lines within the computational domain.

Wave Gauge Name	Spilling (Location [m])	Plunging (Location [m])
WG1	x = 5.945	x = 7.295
WG2	x = 6.665	x = 7.795
WG3	x = 7.275	x = 8.345
WG4	x = 7.885	x = 8.795
WG5	x = 8.495	x = 9.295
WG6	x = 9.11	x = 9.975
WG7	x = 9.725	x = 10.395

4.1.4 Model settings

A relatively long warm up period is necessary to ensure that the volume of water within the domain is constant [Larsen and Fuhrman, 2018]. Following this approach and the findings from Jacobsen et al. [2012], first 60 waves will be discarded from the averaging process. As a result, the total simulation time is about 330 s. This corresponds to approximately > 102 waves in the numerical model. This is the same number of waves as in the experimental setup by Ting and Kirby [1994]. In addition to the previously discussed literature, figure 4.4 from the experimental investigation by Ting and Kirby [1994] also depicts the error in the ensemble averaging as a function of number of waves used in the averaging process. For consistency, 102 waves from the numerical model results will be used to investigate the statistics. A complete listing of the relevant OpenFOAM files can be found in Appendix C.

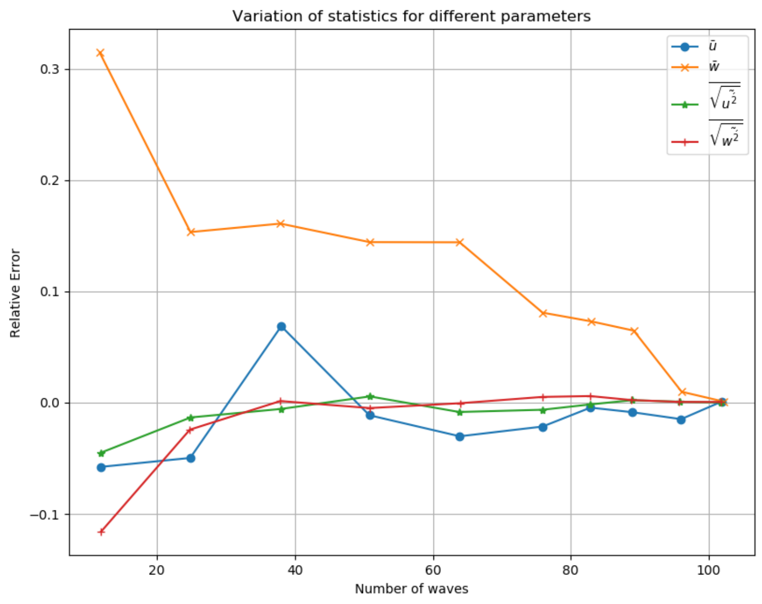


Figure 4.4: Variation of statistics with number of waves [Adapted from Ting and Kirby [1994]]

4.2 EXPERIMENTAL SETUP FOR FLANDERS HYDRAULICS

The wave flume at Flanders Hydraulics Research (FHR) Institute was about 70 m long, 4 m wide and 1.45 m deep (See Figure 4.5). The facility is equipped with a piston-type wave generator with a stroke length of 0.6 m, which is capable of generating regular and random waves with a pre-assigned spectrum. Active wave absorption system (AWAS) was used in the experimental setup which will be under consideration in this particular section. In the experiments, the flume was split into 4 sections as seen in Figure 4.6. Water surface measurements were logged by means of a resistance-type wave gauge, installed at 29 locations within the experimental flume. The sampling frequency of the wave gauge's was 50 Hz. The location of individual wave gauges can be seen in Figure 4.7. Despite the large number of wave gauge's, the data for surface elevation is only available for the wave gauge as listed in Table 4.3. Analysis of the wave height measurements were carried out originally by Altomare et al. [2016] in the time and frequency domain using WaveLab 3.66, software developed at Aalborg University. Classical reflection analysis based on the work done by Mansard and Funke [1980] is based on the linear wave theory which is generally not applicable for very shallow foreshore environments dominated by non-linear effects. Consequently, instead of using this analysis to seperate the incident and reflected wave spectrum, the flume section with no dike (See Figure 4.7) was used to obtain the incident wave spectrum. The presence of cross waves due to the separation baffles within the flume was investigate, however, the difference in wave heights across the flume width was found to be negligible.

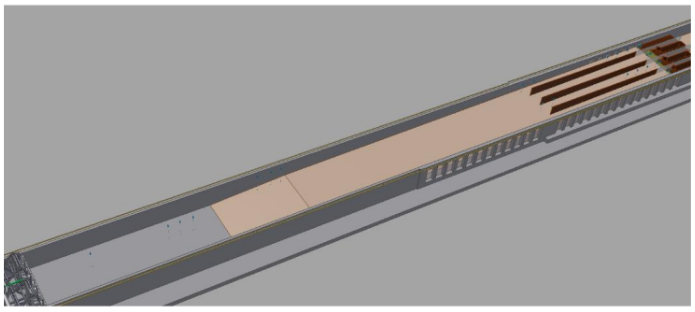


Figure 4.5: Wave Flume at FHR [Altomare et al., 2016]

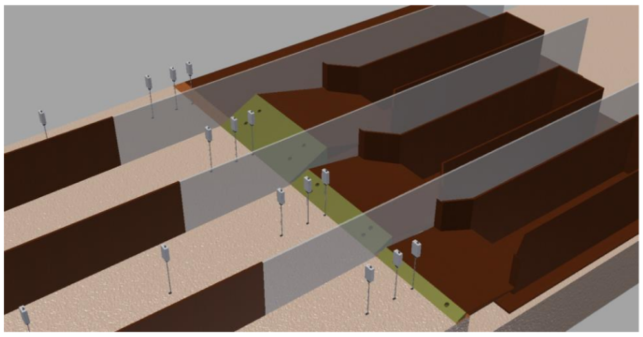


Figure 4.6: Detailed view of the Wave Flume at FHR [Altomare et al., 2016]. The test section which is of interest in this study is the second flume section from the right side of the figure.

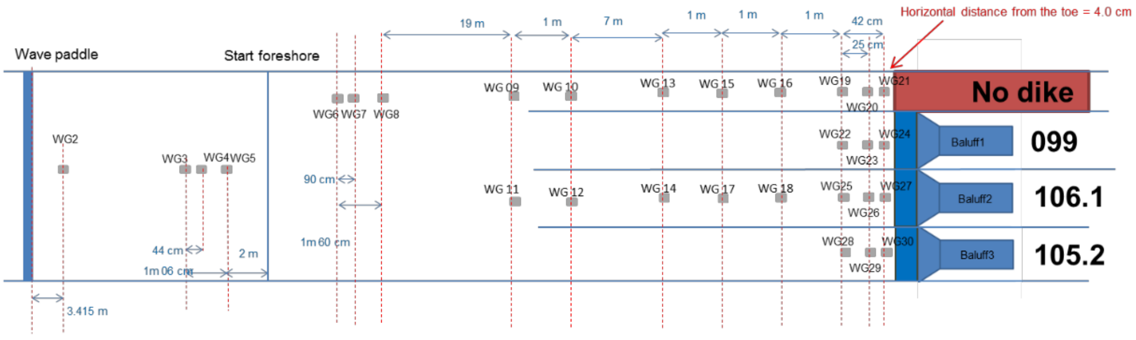


Figure 4.7: Schematic representing the wave gauge locations in the experimental flume. The flume section tagged as 106.1 corresponds to the section which is under investigation in this thesis.

Table 4.3: Location of the wave gauge from the wave paddle in the experimental flume.

Wave Gauge Name	Location [m]
WG2	3.415
WG3	7.94
WG4	8.38
WG5	9
WG6	13.94
WG7	14.84
WG8	15.54
WG25	45.54
WG26	45.79
WG27	45.96

Instantaneous wave overtopping measurements were carried out by two Balluff "Micropulse" water level sensors inside the overtopping boxes. The instrumentation used in the experiments can be seen in Figure 4.8. The instantaneous water level reading from the sensor could be converted to volume by measuring the difference in water level per overtopping wave and multiplying this number by the dimensions of the overtopping box. The Balluff sensors were finally implemented in this project to measure the average overtopping discharge, being related to the difference in water level before and after each test. The wave-by-wave overtopping was not analyzed in detail. The overtopping box dimensions were 0.5 m wide by 2.0 m long and 0.3 m deep. Using this information, the mean overtopping discharge was obtained by dividing the total volume of water collected during the test by the duration of the test $\sim 35 - 40$ min which corresponds to about 1000 waves.

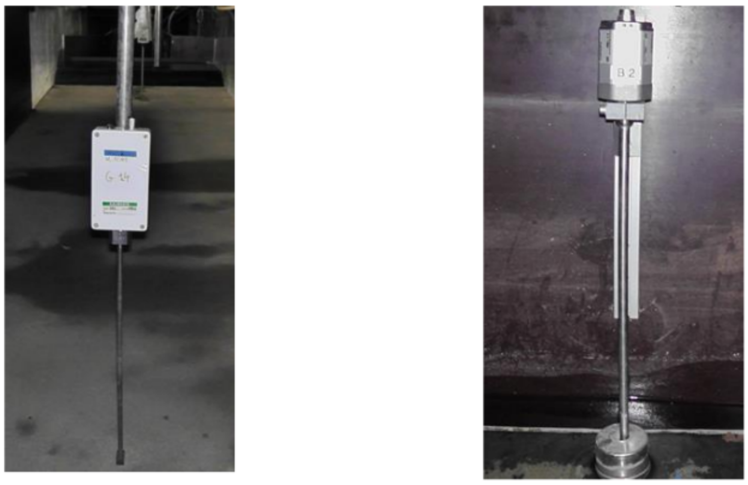
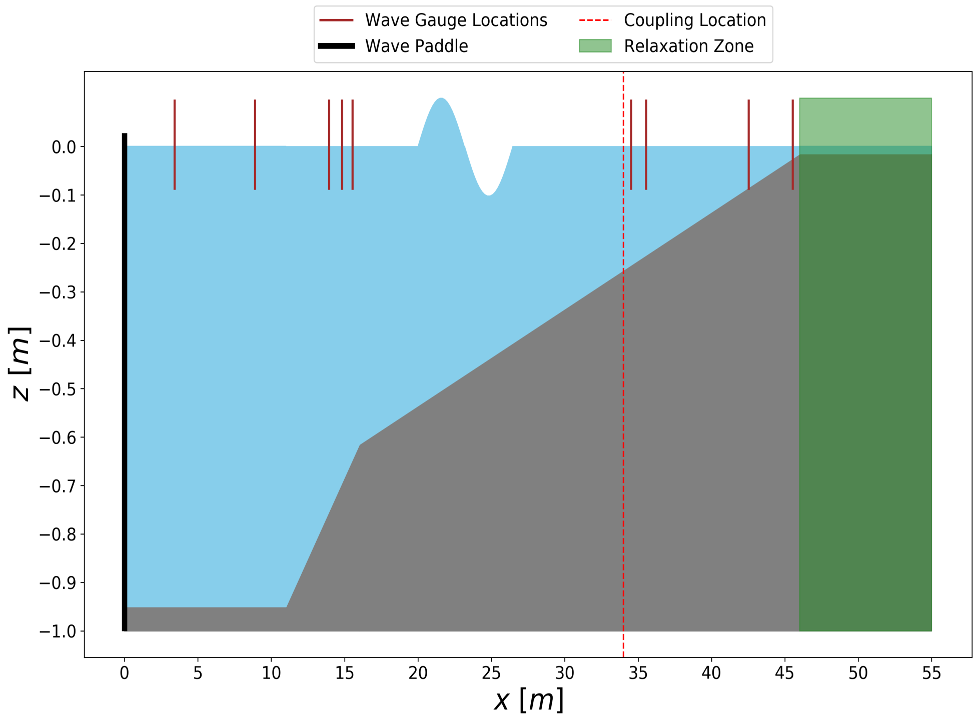


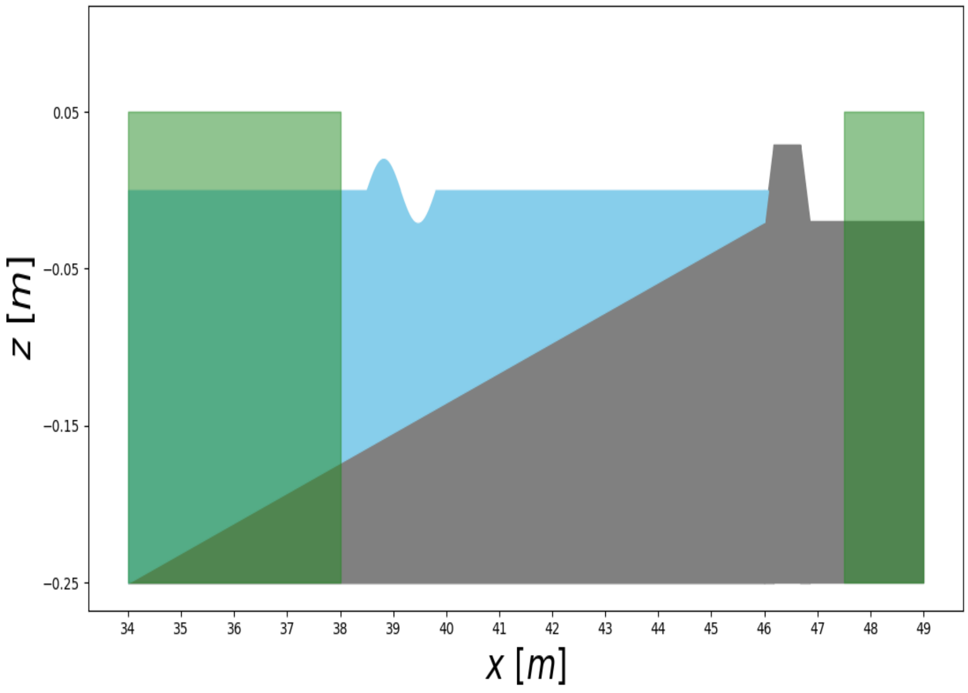
Figure 4.8: Wave gauge (left image) and micro-pulse transducer (right) used to measure the water level in the overtopping box.

4.2.1 Numerical wave tank layout

As seen in Figure 4.5, the length of the domain is quite large in comparison to the previously proposed test case of Ting and Kirby [1994]. Given the large computational time for CFD models, replicating wave flows in such a large domain becomes infeasible. However, Jacobsen et al. [2018] demonstrate the coupling of a potential flow solver named OceanWave3D developed by Engsig-Karup et al. [2009] and OpenFOAM for large domains to estimate hydrodynamics and forces on crest walls. Following a similar approach, the computational domain can be seen in Figure 4.9.



(a) Layout for OceanWave3D model



(b) Layout for the numerical model in OpenFOAM

Figure 4.9: Schematic depicting the computational setup used in the numerical wave tank.

Given the limited availability of the experimental results, a comparison of OceanWave3D (OW3D) and the experimental results will be carried out for *WG*2, *WG*5, *WG*6, *WG*7, *WG*25, and *WG*26 (See Table 4.3). Wave breaking in the experimental setup occurs at around $x = 41$, as a result, the time series comparison of the surface elevation is not directly possible for the wave gauge's in the surf zone. The strategy to validate the numerical model is as follows.

- Compare experimental results with OW3D for wave gauge's $WG2$, $WG5$, $WG6$ and $WG7$. This ensures that the correct offshore wave conditions exist in the numerical wave tank.
- To ensure that the coupling between OpenFOAM and OW3D works correctly, the time series comparison of the surface elevation is carried out at wave gauge's $WG@x = 34.52m$.

This approach makes sure that the surface elevation signal introduced by the wave paddle is correctly transported to the `OpenFOAM` domain by means of the coupling interface between OW3D.

4.2.2 Numerical grid

In order to correctly model the coupled system, the numerical grid in both the models should be able to adequately capture the incident wave conditions. Consequently, the numerical grid in the offshore domain (OceanWave3D) was discretized as seen in Figure 4.10 with a $dx \sim 0.026m$. In order to better capture the vertical velocity profile, vertical stretching has been used for the grid which can be seen in the figure.

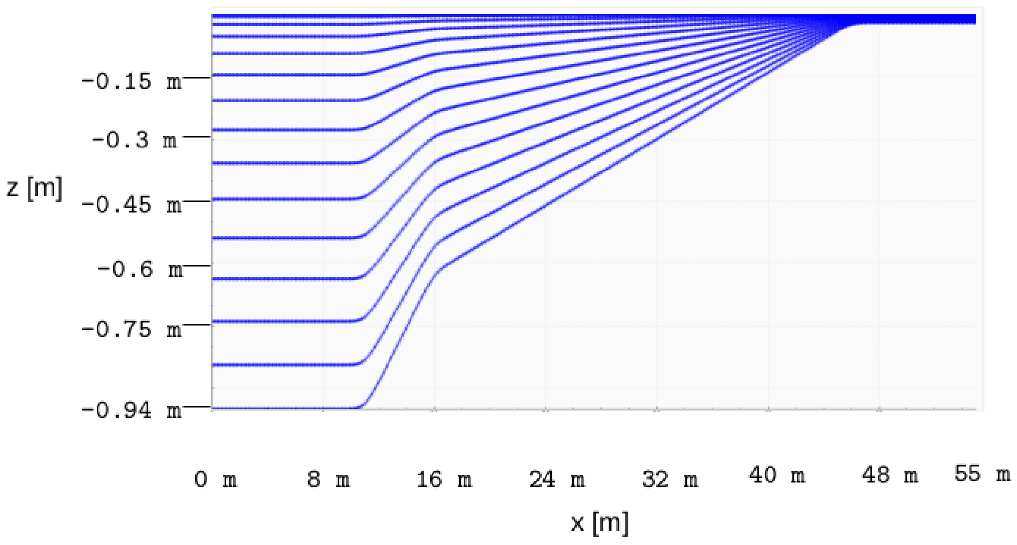


Figure 4.10: Schematic of the numerical grid used in OceanWave3D. The different lines represent the layer used in the vertical direction to solve for the momentum in Z axis.

The time step used in the OceanWave3D setup was $dt = 0.002s$. The `OpenFOAM` grid on the other hand used a finer grid definition with a resolution of $dx \sim 0.0068m$. The grid was generated using a high quality Finite Element Mesh generation tool named Gmsh Geuzaine and Remacle [2009]. The grid was generated in such a way as to keep a consistent aspect ratio of ~ 1 in the foreshore region of the dike based on the discussions in Jacobsen et al. [2012]. The reason for using Gmsh over the conventional `blockMesh` tool within `OpenFOAM` was the ability of Gmsh to produce smooth boundary conforming mesh in regions where the change in the slope is drastic. For example, the connection between the dike and the foreshore would result in relatively skewed grid cells if the blocking strategy used is not good. However, with limited effort Gmsh seems to provide a faster solution for grid generation for such cases. Consequently, this was preferred over the conventional approach[1].

4.2.3 Model Settings

The model setup for this case is identical to the technique used in Section 4.1.4. In order to compute the initial turbulence levels, the significant wave height and wave period was used. Using this information, the formulations discussed for the previous test case setup were used to arrive at the initial conditions for the turbulence model. Since this case has fundamentally different incident wave conditions in comparison to the previous model setup, wave generation in OceanWave3D was carried out as described in the sub-section below.

A wide variety of options for specifying the incident wave conditions were available for however, in order to prescribe the exact wave conditions as in the experimental investigations by Altomare et al. [2016], the wave paddle location recorded during the experiment was used. OceanWave3D requires a wave paddle velocity signal in order to produce the wave conditions, consequently,

$$v(t) = \frac{dx(t)}{dt}, \tag{4.2}$$

was used to arrive at the wave paddle velocity. In order have better precision, central differencing in the numerical derivative was used with appropriate handling at the end points. This wave paddle velocity was observed to provide under-prediction in the initial test runs as a result, the velocity signal was amplified by 5% to get the correct wave signal at the offshore boundary. In addition to this amplification, the fixed sampling frequency for the wave paddle location and the subsequent derivative results in a 'spikey' velocity signal. In order to smooth out these small fluctuations, a Savitzky-Golay filter as discussed in Persson and Strang [2003]. The difference between the filtered and the unfiltered signal can be seen in Figure 4.11.

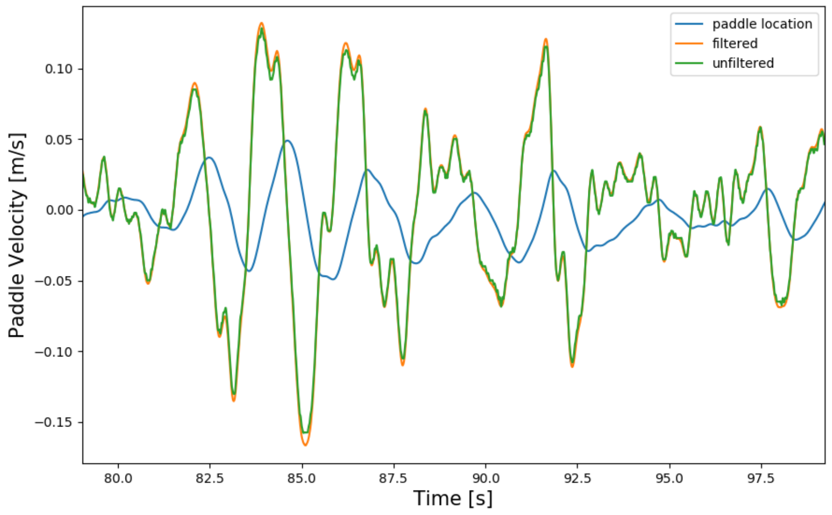


Figure 4.11: Paddle velocity prescription in the numerical model.

[1]The author however does acknowledge the fact that with a good blocking strategy, `blockMesh` utility would also yield a satisfactory grid.

5 RESULTS AND DISCUSSIONS

OUTLINE

This chapter presents the results from each of the numerical investigations carried out in the thesis. The very first section deals with the Ting and Kirby [1994] experimental dataset which will aim to establish a benchmark for the new numerical model setup. This chapter will discuss fundamental regular wave conditions along with differences between spilling and plunging breaker hydrodynamics. The final section of this chapter will explore the Flanders hydraulics experimental setup for irregular waves and overtopping behavior.

5.1 NUMERICAL RESULTS FOR TING AND KIRBY [1994] DATASET

The results for Ting and Kirby [1994] experiments have been simultaneously compared with the studies carried out by Larsen and Fuhrman [2018]. The following sections detail comparison of the following key parameters which will be presented and discussed in the section:

- Surface elevation
- Turbulent kinetic energy profiles (TKE profiles)
- Velocity structure (undertow)

5.1.1 Surface elevation

In the experimental campaign by Ting and Kirby [1994], the mean, maximum, and minimum surface elevation were sampled for the flume coordinates $x \sim -2$ m to $x \sim 12$ m. These surface elevation profiles were ensemble averaged for about 102 waves in the experimental studies at each wave gauge (See Ting and Kirby [1994]). A similar comparison was also carried out with the numerical model. As prescribed by Jacobsen et al. [2012], Larsen and Fuhrman [2018], and Larsen et al. [2018] the wave simulations should be preceded by atleast 30 wave periods before any statistical analysis can be carried out. Consequently the first 60 s in the spilling breaker case and the first 150 s in the plunging breaker case were discarded from the analysis presented in the following section.

It was observed that reflection of the incident waves introduced some variations in the waves outside the relaxation zone in the numerical model. This behavior can be seen in Figure 5.1. In order to compute the maximum and minimum for the given time series, a basic filter window was applied to locate the respective maximum and minimum. The python script used to carry out the analysis has been presented in Appendix section D.

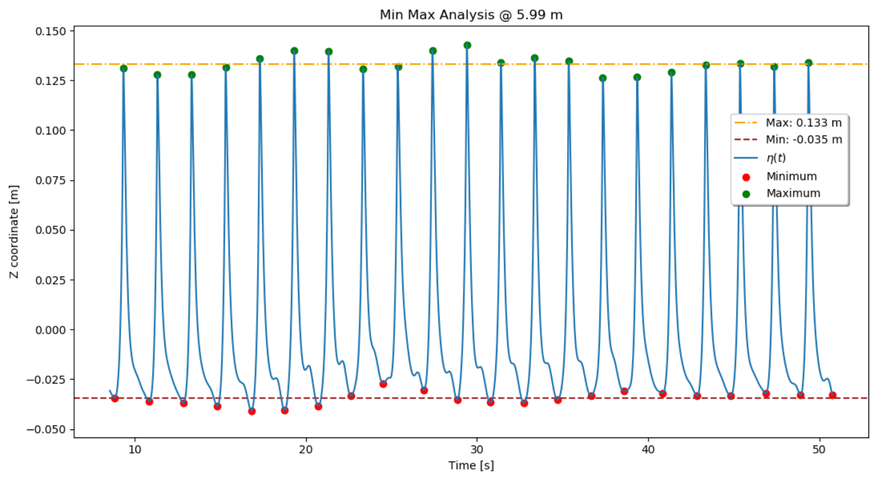


Figure 5.1: Surface elevation of the wave gauge located at $x \sim 5.99$ m in the numerical model. The surface elevation time series corresponds to the spilling wave breaker from Ting and Kirby [1994]. The figure also depicts the locations of maximum and minimum points for each wave period.

There is a systematic difference between the ensemble average method used in Ting and Kirby [1994] and the method used to obtain the average min, max, and mean profiles as described above in Figure 5.1. The technique used in this analysis basically locates the maximum or minimum based on the hard-coded window. This may introduce a bias to the higher and the lower values for given time series. Phase averaging would involve overlaying each wave cycle over each other and then averaging the values, thus possibly reducing the observed average surface elevation. However, it is expected to not significantly change the statistics as presented in this analysis. As a result, this method was used to carry out the wave statistics instead of using a phase averaged approach.

Spilling breaker

Figure 5.2 shows the $\langle\eta_{max}\rangle - \langle\overline{\eta}\rangle$, $\langle\overline{\eta}\rangle$, and $\langle\eta_{min}\rangle - \langle\overline{\eta}\rangle$ profiles for the spilling breaker case. The shaded region around the maximum and the minimum profiles represent one standard deviation envelopes on either sides of the average/mean value. The mean surface elevation profile is captured sufficiently well prior to the surf zone. The mean surface elevation, TKE, and undertow profiles from Larsen and Fuhrman [2018] are compared for the stabilization coefficients $\lambda_1 = 0.875$ and $\lambda_2 = 0.05$, which has been presented in Figure 5.2.

The numerical model used in this investigations captures the wave breaking height and location sufficiently well. In comparison to the previous studies, the *waveFlow* solver does not overpredict the wave height and the wave breaking location approximately more than one standard deviation (marked by the orange envelope around the black line). It was observed that the wave setup undergoes a delay after the breaking point. This can be attributed to the roller transported after wave breaking occurs (at around x $\sim$ 6.4 m) while the wave setup in the numerical model is observed to initiate only at about x $\sim$ 8 m. This behavior has also been observed in the studies carried out by Larsen and Fuhrman [2018]. Roelvink and Stive [1989] and Svendsen [1984] suggest that this delay is attributed to the fact that the turbulent kinetic energy is not immediately dissipated after breaking. A detailed analysis of this phase lag between the wave setup and the wave breaking location was also carried out by Jacobsen et al. [2014]. In this study Jacobsen et al. [2014] pointed out that there is a definitive relationship between the non-dimensional lag and the surf similarity parameter which governs wave breaking type. This relationship is given by the expression as described in Equation 5.1,

$$\lambda^*_{HS} = 0.25\zeta_0^{-0.7} \tag{5.1}$$

where $\lambda^*_{HS} \sim (x_H - x_{\overline{\eta}})/(T\sqrt{gh_B})$ is the non dimensional phase lag, x_H is the wave breaking location [m], $x_{\overline{\eta}}$ is the location at which wave setup initiates [m], T is the wave period [s], g is the gravitational acceleration [m/s^2], and h_B is the local depth. The behavior for the two tests considered in this investigation can be seen in Figure 5.3.

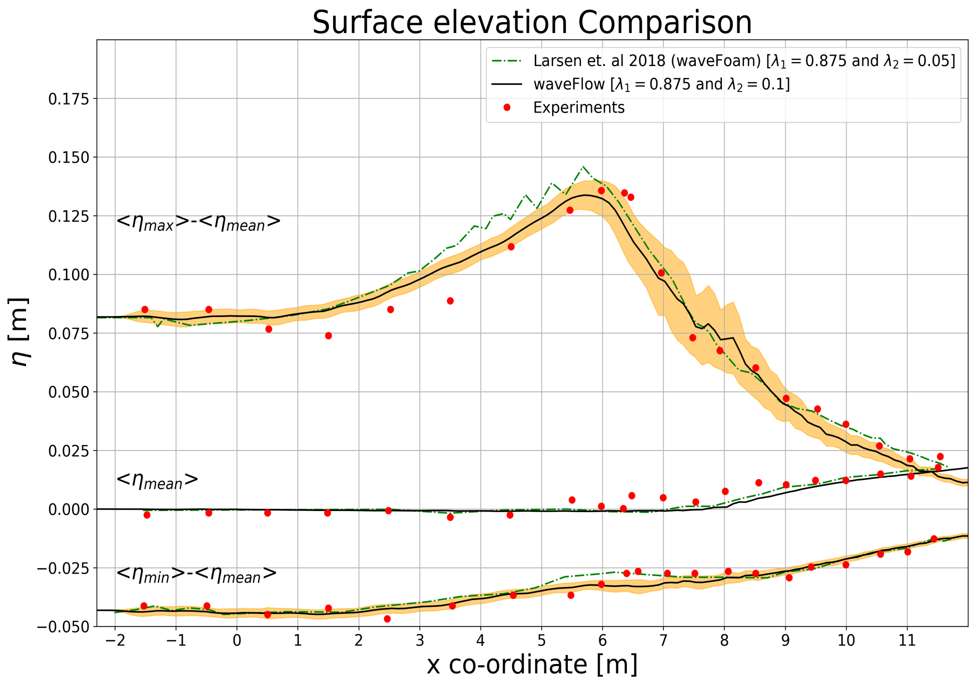


Figure 5.2: Comparison of ensemble averaged surface elevation profiles for the spilling breaker case of Ting and Kirby [1994].

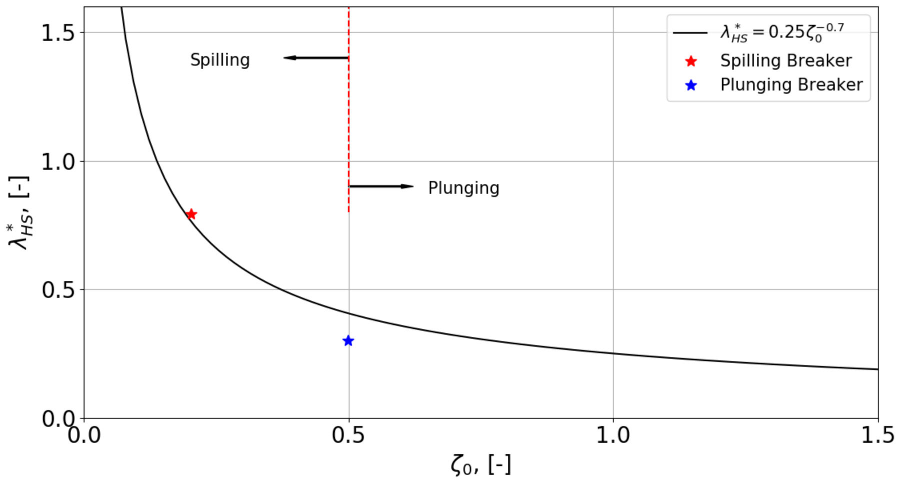


Figure 5.3: Comparison of the non dimensional phase lag λ^*_{HS} as a function of ζ_0 as discussed by Jacobsen et al. [2014].

The effect of the roller transported after wave breaking can also be seen in the large standard deviation in the maximum surface elevation plotted ($\langle\eta_{max}\rangle - \langle\overline{\eta}\rangle$)

in Figure 5.2. The large variability in the maximum surface elevation depicts the roller which results in wave-wave variability in the surf zone, thus resulting in a larger standard deviation within the surf zone. Despite the large variability, the general trend pretty much follows the experiments investigations from Ting and Kirby [1994].

Plunging breaker

Figure 5.4 shows the $\langle\eta_{max}\rangle - \langle\overline{\eta}\rangle$, $\langle\overline{\eta}\rangle$, and $\langle\eta_{min}\rangle - \langle\overline{\eta}\rangle$ profiles for the plunging breaker case. The shaded envelope holds the same meaning as discussed in the previous sub-section. The results from Larsen [2018] have been presented for the same stabilization coefficients for consistent comparison.

As observed for the spilling case, the wave breaking height and the wave breaking location in the plunging breaker case is captured sufficiently accurately. The standard deviation reduces close to the breaking point since the wave-wave variability close to the breaking point is almost negligible, this can be seen in Figure 5.5. As observed in the spilling case, the transported roller after wave breaking introduces wave-wave variability within the surf-zone in the plunging wave breaker. A delayed wave setup can be consequently observed in both the numerical models seen in Figure 5.4. Considerable differences were observed in the minimum surface elevation ($\langle\eta_{min}\rangle - \langle\overline{\eta}\rangle$) within the surf zone.

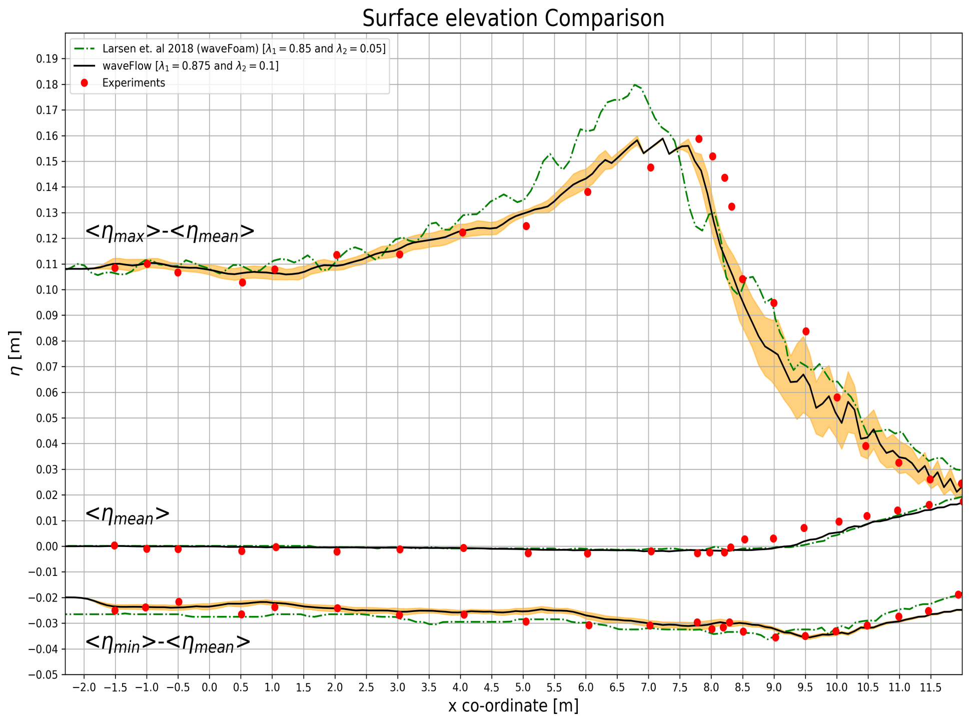


Figure 5.4: Comparison of ensemble averaged surface elevation profiles for the plunging breaker case of Ting and Kirby [1994].

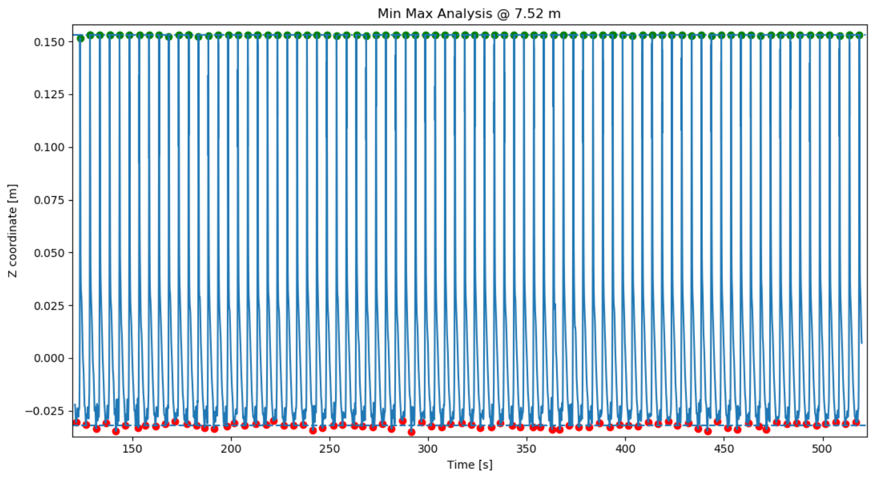


Figure 5.5: Surface elevation close to the breaking point in the numerical model for plunging wave breaker.

5.1.2 Turbulent kinetic energy profiles (TKE profiles)

In order to successfully compare the performance of the stabilized turbulence models for long term wave simulations, a comparison of the turbulence kinetic energy (TKE) within the surf zone has been performed. Suitable time averaging technique has been used to evaluate the distribution of TKE within the water column. For practical reasons, the TKE profiles in the numerical model have only been compared under the trough level. Suitable time averages were used in order to arrive at a time averaged TKE profile. About 7 wave gauge's were placed within the water column to sample the results every 0.1 s (or 10Hz frequency).

Spilling Breaker

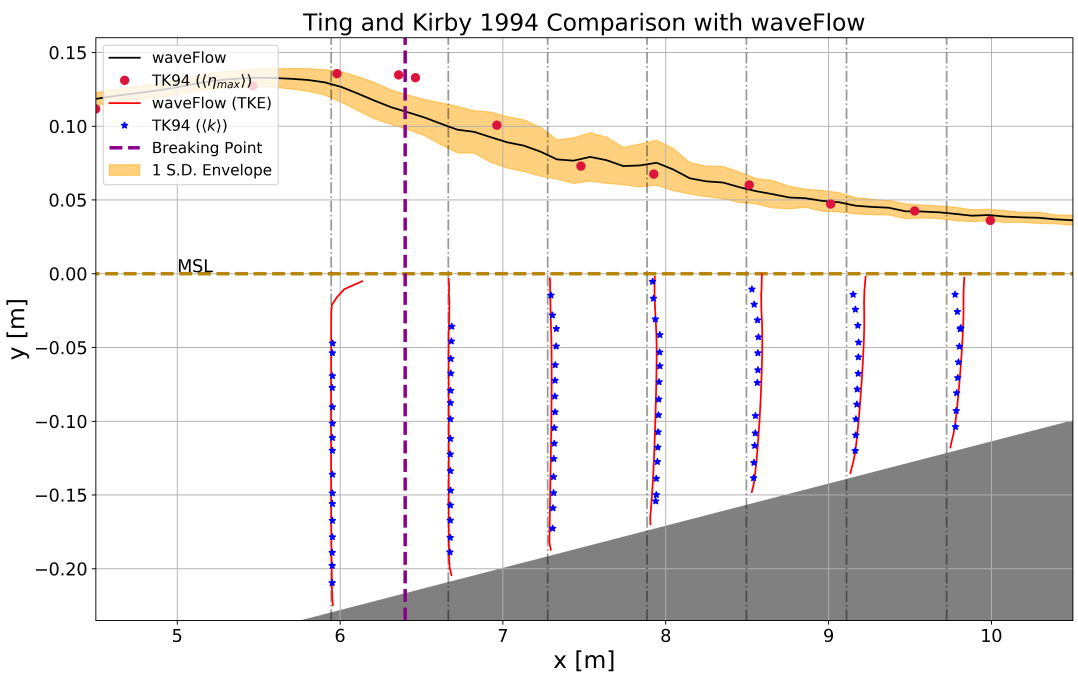
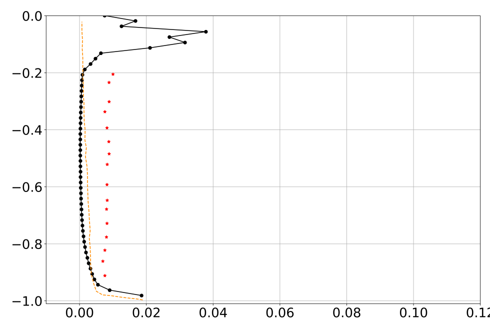
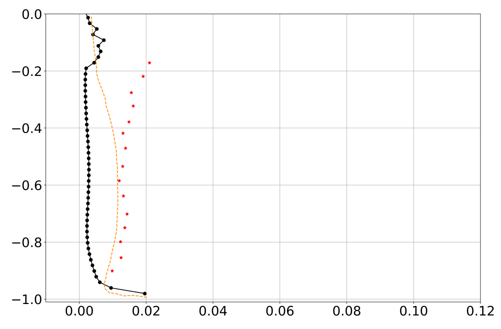


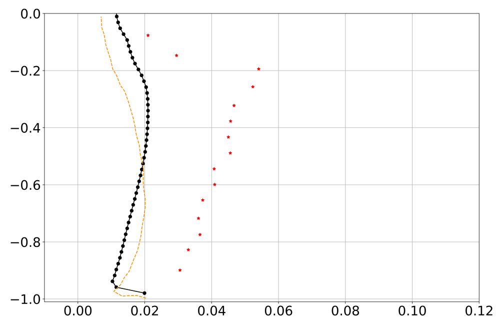
Figure 5.6: Turbulent Kinetic Energy profile comparison for spilling breaker case.

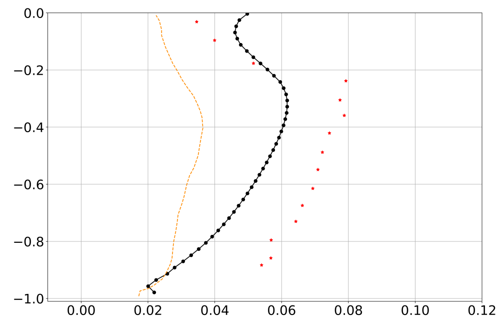
In the Figure 5.6, the red filled circles correspond to the experimental surface elevation measurements from Ting and Kirby [1994]. The dark black lines and the orange envelope represent the numerical model results and the corresponding standard deviation envelope around the mean value. The blue stars represent the experimental measurements for the TKE at the wave gauge locations as described in Table 4.2. The red solid line corresponds to the numerical prediction of the TKE values at the given wave gauge locations which are represented by the dotted vertical grey lines. As seen in the above figure, the TKE profiles seem to represent the experimental trends satisfactorily. Consequently, the stabilization parameters as used in this investigation provide satisfactory results both in the prediction of TKE and mean surface elevation. Figure 5.6 provides an overview of the TKE distribution for the spilling breaker.

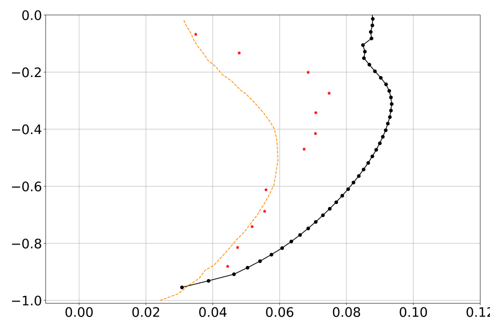
Figure 5.7 shows the comparison of the numerical results for the wave gauge's within the surf zone. A gradual overprediction is observed as we move deeper within the surf zone. Although the results vary drastically, they are consistent with the choice of λ_1 and λ_2 parameters in this thesis work. The overprediction is attributed to the $\lambda_2 = 0.1$ value. As discussed in Section 2.5.3, the role of this stabilization coefficient is to allow the filter to switch on for a larger allowable ratio of p_Ω / p_0. Consequently, a larger generation of TKE in the potential flow region is allowed which is directly reflected in the numerical results. Since, Larsen and Fuhrman [2018] use a value of 0.05 for this coefficient, the TKE obtained at the same location is relatively smaller and conforms to the experimental results. It is also important to note that the TKE profiles are weighted averaged with the alpha fraction. This is done in order to avoid including the TKE values for the air phase. Since the air phase in the current VOF method also has a certain TKE value, including the results for this phase would return erroneous values above the trough level. This process also leads to a large fluctuation closer to the region above the trough level. This can be seen in the figures below.

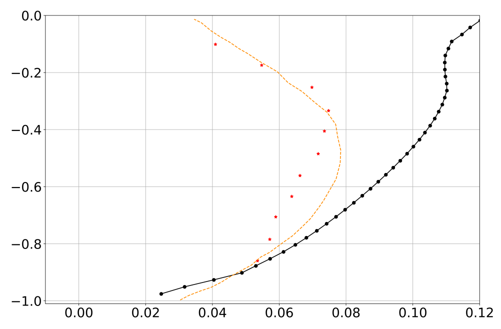

(a) WG1

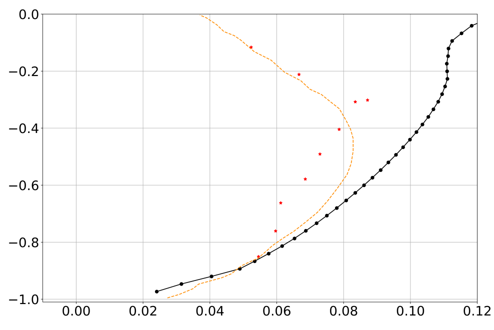

(b) WG2


(c) WG3


(d) WG4


(e) WG5


(f) WG6


(g) WG7

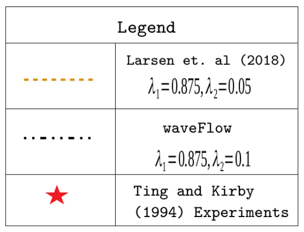


Figure 5.7: Comparison of TKE profile against the experimental results for the spilling breaker. The x-axis denotes $\sqrt{(\langle k \rangle / (gh))}$ while the y-axis represents $(z - \langle \eta \rangle)/h$. The black line represents the current study, while the orange dotted line corresponds to the results by Larsen and Fuhrman [2018] and the red stars correspond to the experimental results by Ting and Kirby [1994].

Plunging Breaker

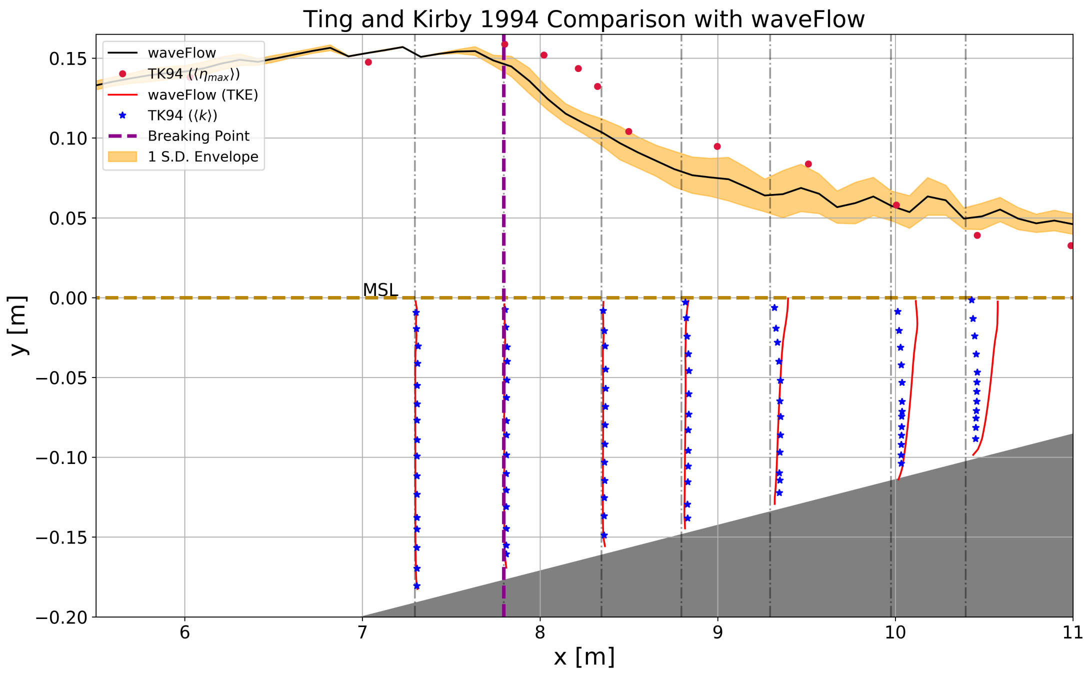
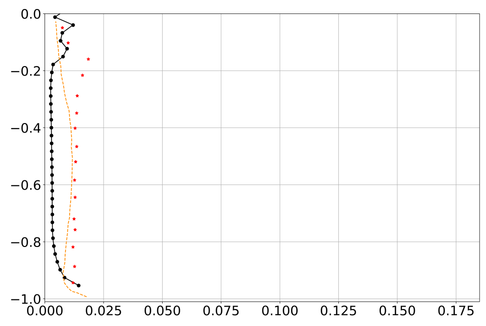
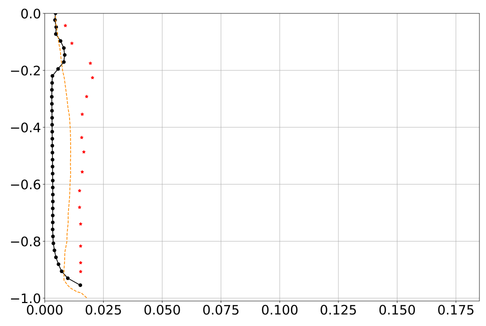


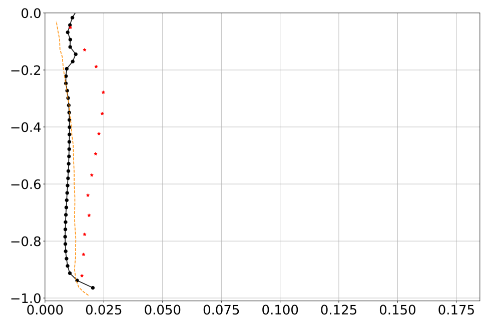
Figure 5.8: Turbulent Kinetic Energy profile comparison for plunging breaker case.

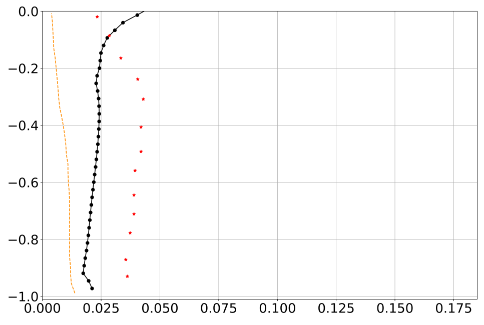
Similar results have been observed in the plunging breaker case. The undertow profiles close to the breaking point in the surf zone follow the experimental results satisfactorily. There seems to be an over-prediction in the turbulence levels deep inside the surf-zone. However, this over-prediction is in-line with the results obtained by [Larsen and Fuhrman, 2018]. These results show that new turbulence closure results in bounded values for the turbulent kinetic energy post wave-breaking. In comparison to the un-bounded turbulence kinetic energy results as seen in [Pengzhi and Philip, 1998] and [Hsu et al., 2015] to list a few.

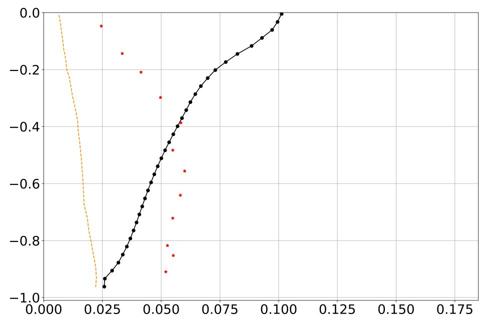
Individual result comparison with the experimental TKE profiles reveals a similar trend as observed in the spilling breaker case. The choice of λ_2 parameter affects the TKE profile resolution. The results obtained by Larsen and Fuhrman [2018] are in agreement with the TKE profiles observed in Figure 5.9. Within the framework of the new turbulence closure the results for TKE are bounded yet could be improved especially deep inside the surf zone.

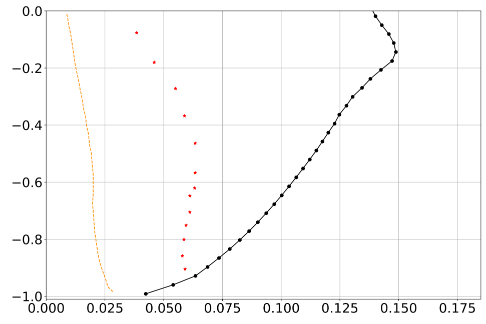

(a) WG1

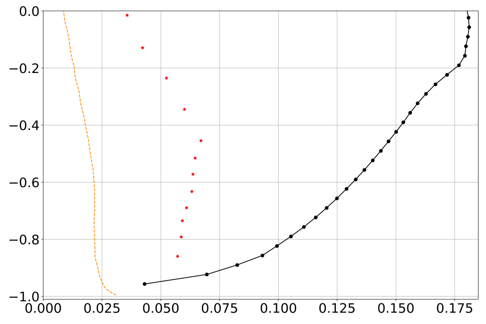

(b) WG2


(c) WG3


(d) WG4


(e) WG5


(f) WG6


(g) WG7

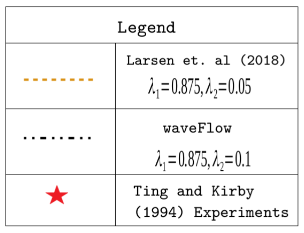

Legend
Larsen et. al (2018)
λ_1=0.875,λ_2=0.05
waveFlow
λ_1=0.875,λ_2=0.1
Ting and Kirby (1994) Experiments

Figure 5.9: Comparison of TKE profile against the experimental results for the plunging breaker. The x-axis denotes $\sqrt{(\langle k\rangle/(gh))}$ while the y-axis represents $(z-\langle\eta\rangle)/h$. The black line represents the current study, while the orange dotted line corresponds to the results by Larsen and Fuhrman [2018] and the red stars correspond to the experimental results by Ting and Kirby [1994].

5.1.3 Velocity Structure

The velocity sampling followed a similar sampling approach with a frequency of 10 Hz. Averaging techniques similar to the ones used in the previous section were used in order to compute the time averaged undertow profiles. The sub-sections below present the results for the time averaged undertow profiles for the spilling and the plunging wave breakers.

Spilling Breaker

As seen in Figure 5.10, the undertow profiles have been adequately captured in the numerical model. Barring some small discrepancies between the experimental and the numerical results in the bottom part of the undertow profile, the general structure of the undertow has been substantially well represented in this case. However, as opposed to experimental observations the top part of the velocity profile seems to be overpredicted in the numerical model. Additionally, the maximum shoreward velocity is not located at the SWL. These discrepancies in the upper part of the velocity profile are not observed in the results obtained by Larsen and Fuhrman [2018]. At the same time, they do not present the complete velocity profile above the trough level, thus a direct comparison is not possible. The velocity profile seems to improve with regards to the location of the shoreward maximum as we go deeper into the surf zone after the breaking point. However, the discrepancy between the numerical and the experimental results increases in the bottom part of the profile. An associated higher velocity is also observed in the top portion of the velocity profile which balances the overshoot in the bottom. The results in Figure 5.10 perform relatively poorly in the regions where the TKE prediction is overshooting in the numerical model. There seems to be a definite relationship between the prediction of TKE profile and the velocity resolution within the surf zone.

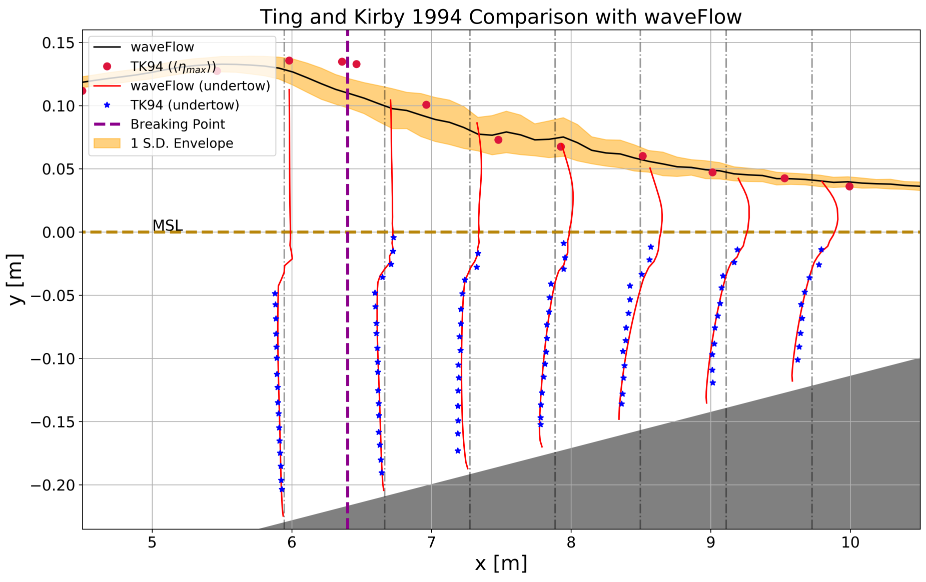


Figure 5.10: Undertow profile comparison for spilling breaker case.

One to one comparison of the undertow profile has been presented in Figure 5.11. There is a significant under-prediction in the bottom part of the velocity profile in comparison to the results by Larsen and Fuhrman [2018]. This difference is attributed to the different λ_2 parameters used in this study. However, no fundamental improvements have been observed in the general trend of the undertow profile.

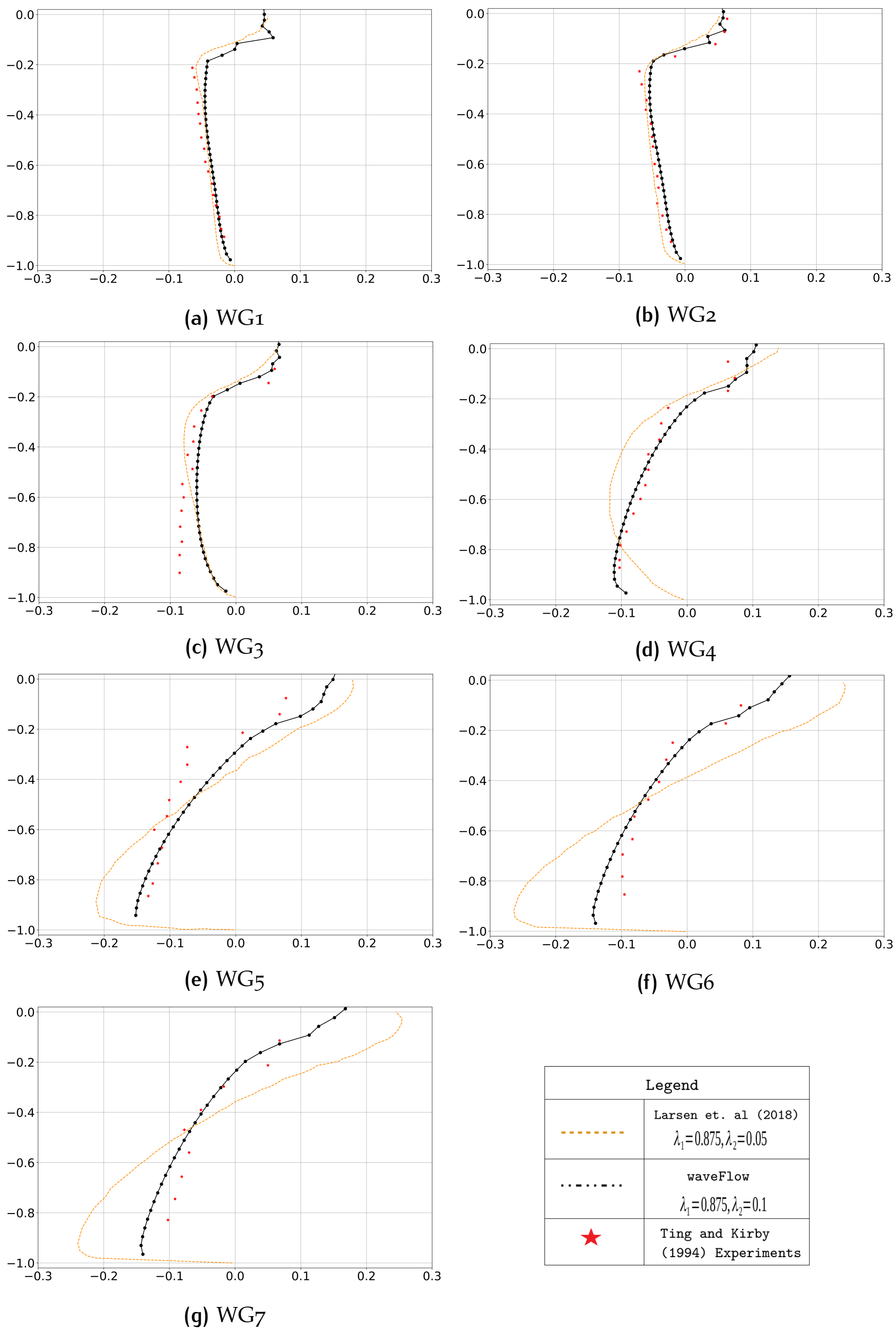


Figure 5.11: Comparison of undertow profile against the experimental results for the spilling breaker. The x-axis denotes $\langle u \rangle / (\sqrt{gh})$ while the y-axis represents $(z - \langle \eta \rangle)/h$. The black line represents the current study, while the orange dotted line corresponds to the results by Larsen and Fuhrman [2018] and the red stars correspond to the experimental results by Ting and Kirby [1994].

Plunging Breaker

As seen in Figure 5.12, the undertow profile suffers an over-prediction in the same surf-zone regions where the turbulent kinetic energy is over-predicted. In the regions close to wave breaking, the undertow profile has been quite accurately captured in the numerical model. The discrepancies in the undertow profile seem to be increasing with increasing x coordinate i.e. deeper within the surf-zone. A similar trend is observed in the spilling case. As discussed before this shows the relation between the velocity profile and the TKE profile. Using the same algorithm as the spilling case for computing the mean velocity profile, it is observed that there is no overshoot in the numerical prediction of the velocity profile before wave breaking. However, deeper in the surf-zone the bottom part of the velocity profile starts to overpredict the velocity magnitude and a corresponding increase is observed in the upper part of the velocity profile. In general, as there are no experimental data points in the region above the trough level, a direct comparison is not possible. However, the maximum shoreward velocity is observed to be around the SWL unlike the spilling case.

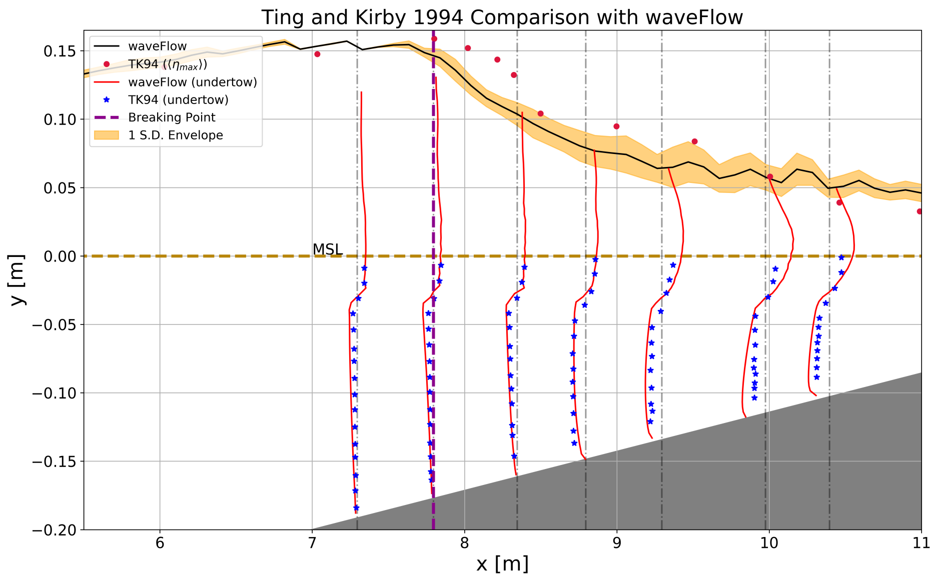
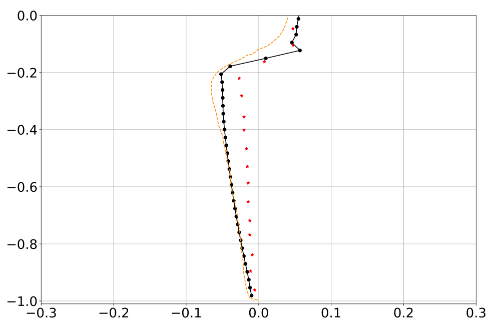
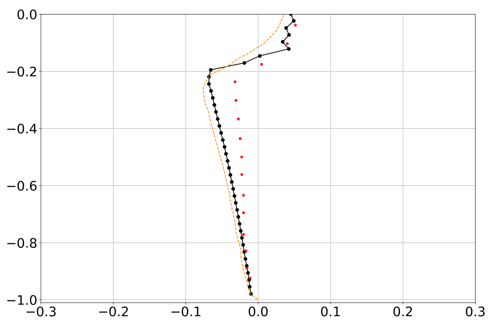


Figure 5.12: Undertow profile comparison for plunging breaker case.

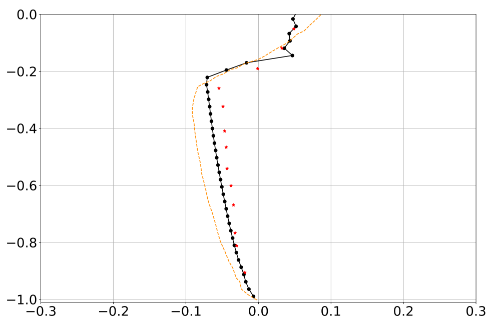
Comparable results are obtained in the one to one comparison of the velocity profile. Similar argument seems to hold in terms of the velocity prediction and the TKE prediction for the numerical model. In general, the bottom part of the velocity profile seems to be overpredicted and consequently the top part of the velocity profile is also overpredicted. This is a direct consequence of mass flux balance. A vertically integrated profile would yield zero effective velocity. Given that the experimental and numerical data are one dimensional and not of the same size, any statistical tests on the comparison have not been carried out.



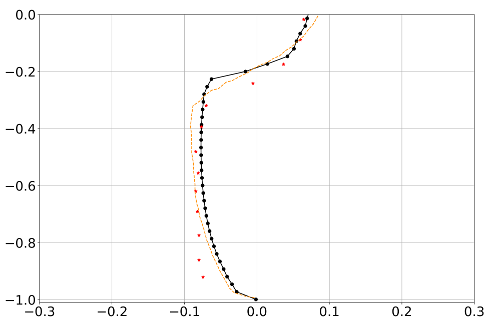
(a) WG1



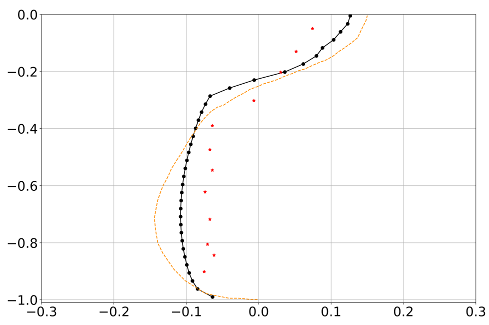
(b) WG2



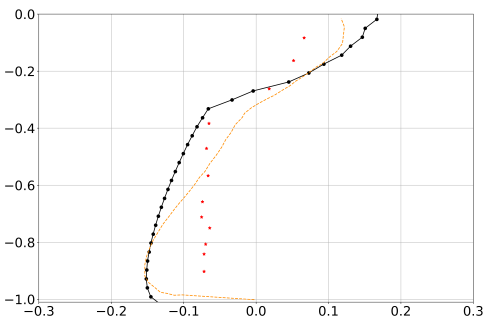
(c) WG3



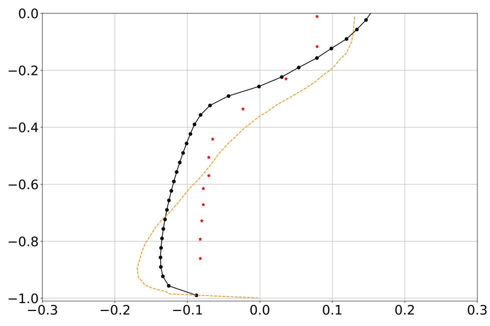
(d) WG4



(e) WG5



(f) WG6



(g) WG7

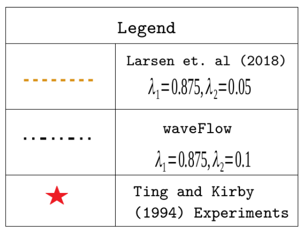

Legend
Larsen et. al (2018)
λ_1=0.875, λ_2=0.05
waveFlow
λ_1=0.875, λ_2=0.1
Ting and Kirby (1994) Experiments

Figure 5.13: Comparison of undertow profile against the experimental results for the plunging breaker. The x-axis denotes $\langle u \rangle / (\sqrt{gh})$ while the y-axis represents $(z - \langle \eta \rangle)/h$. The black line represents the current study, while the orange dotted line corresponds to the results by Larsen and Fuhrman [2018] and the red stars correspond to the experimental results by Ting and Kirby [1994].

5.2 NUMERICAL RESULTS FOR FLANDERS EXPERIMENTS

As discussed in Section 4.2.1, the lack of data available throughout the flume domain for the Flanders case makes a direct comparison of the results cumbersome.

However, as described in the same section, comparing the results individually in different sections of the flume can provide substantial confidence that the coupled modeling strategy is working effectively. Consequently, the results will be discussed as listed below

- Surface elevation comparison of OceanWave3D and Experimental results in the flume section $x < 34$ m. This ensures that right offshore wave conditions are imposed in the numerical wave tank.
- Surface elevation comparison of OceanWave3D and `OpenFOAM` for two wave gauges located at $x \sim 34.52$ m and $x \sim 35.54$ m. This ensures that the coupling between the two models is working as expected.
- Overtopping results for the coarse mesh used in this study.

The spectral development over the flume bathymetry for the given case can be seen in Figure 5.14. The spectrum shown in this figure correspond to the experimental results. The black line plotted over the left vertical plane corresponds to the bathymetry of the flume. The figure shows a shift in the peak frequency from the high frequency side to the low frequency side beyond the wave breaking location ($x \sim 39 - 40m$). The figure also depicts the spectral evolution inside the swash zone where significant differences can be seen between the spectrum. There seems to be a large shift in the spectral energy towards the higher frequency between the last two wave gauges within the flume. However, a detailed analysis is required in order to identify the exact source of this shift.

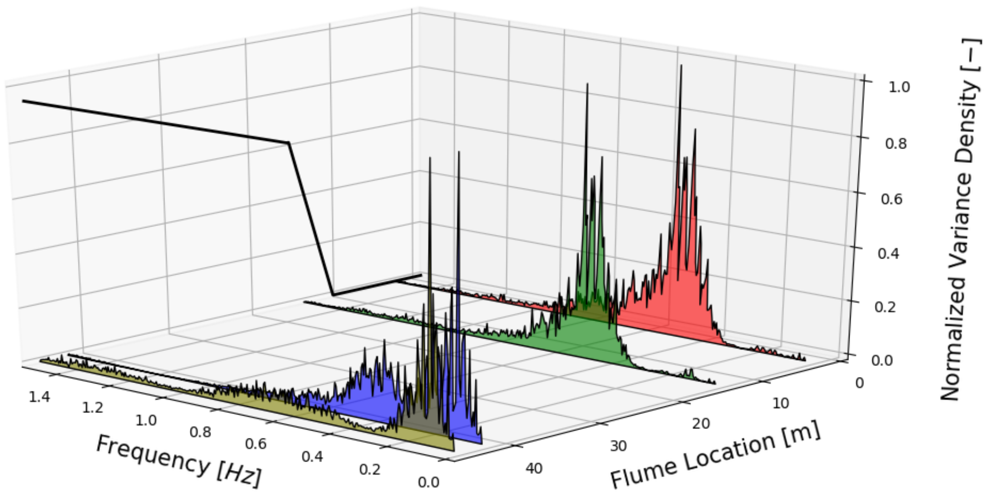


Figure 5.14: Spectral evolution over the flume bathymetry for the Flanders experimental campaign. The waves are incident at flume location $\sim 0m$. The spectral plots correspond to the experimental results.

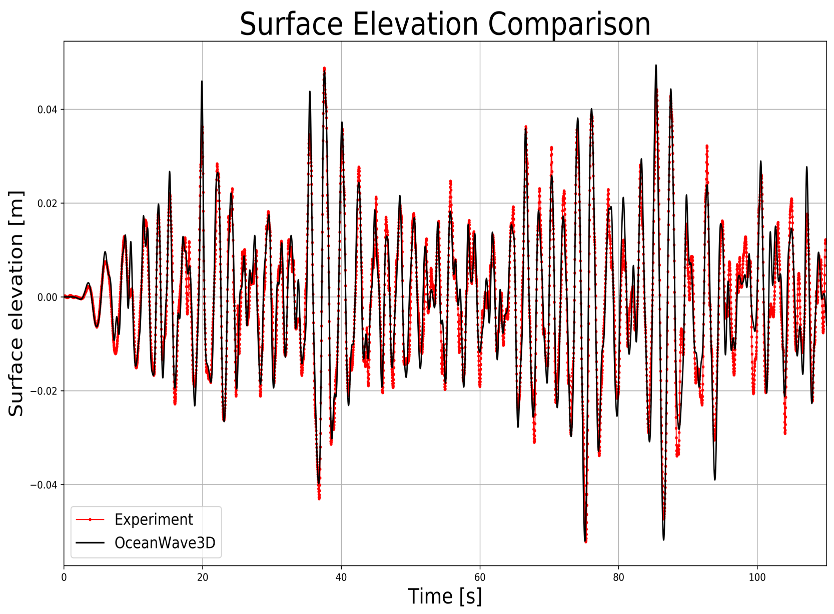
5.2.1 Surface elevation comparison of OW3D and Experiment

In order to compare the experimental results, the analyzed time series obtained from Altomare et al. [2016] has been used. The first section of this discussion compares the time series of the surface elevation along with the variance density spectrum for the given time slice. As seen in Figure's 5.15, 5.16, 5.17, and 5.18, the time series as well as the frequency comparison shows good replication of incident wave conditions in the numerical wave tank. The potential flow solvers seems to be performing quite well for relatively large waves, however, the smaller waves in a given wave group are sometimes not resolved adequately. However, this is acceptable since the most interesting part of the wave group are the larger waves which results in an overtopping event which is substantial. The frequency space shows a significantly different peak behavior around the peak frequency for the wave gauge's closer to the wave paddle (See Figure's 5.15 and 5.16). Despite this behavior the peak frequency even with the small time domain is correctly located at ~ 0.4425 *Hz* which corresponds to the peak period of ~ 2.25 s in the numerical model. The experimental spectrum shows a consistent multiple peak behavior where the two (or more) peaks have similar energy.

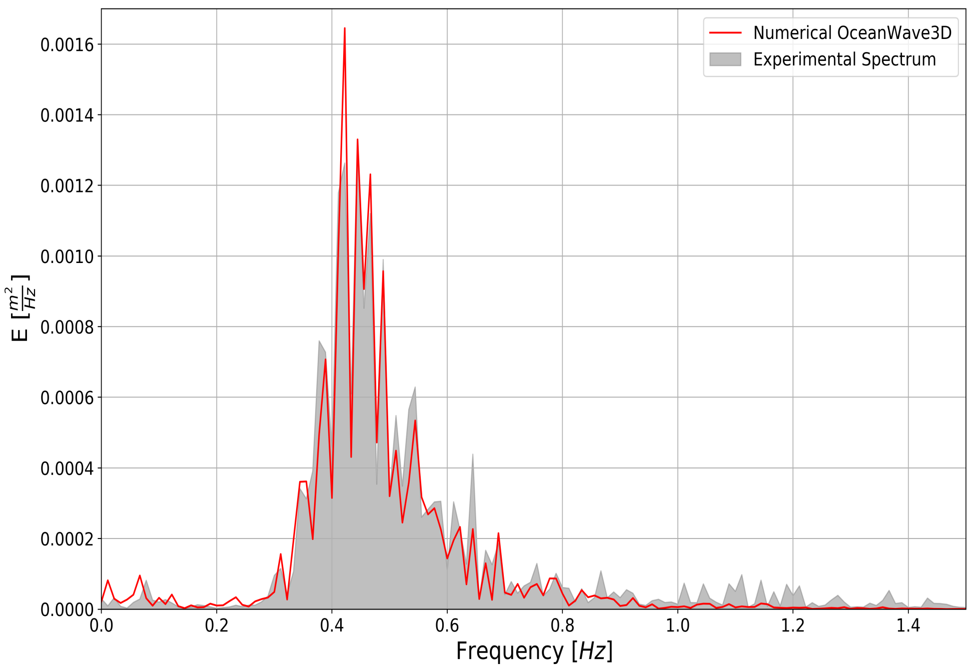
In order to compute the variance density spectrum, it is critical to note that the sampling frequency in the OW3D model is non-constant. As a result, carrying out a normal Fourier transform is not possible. In this particular case, computing the non-uniform Fourier transform also does not yield adequately comparable results. Considering this issue, a practical approach was used to employ the mean sampling rate for the given time series of the numerical surface elevation. This approach gives a decent comparison of the wave spectrum throughout the analysis, as a result, it was used to obtain the numerical variance density spectrum.

As discussed in the model setup (See Section 4.2.3), an amplification of 5% was used to input the right wave amplitude in the numerical wave tank. This along with the approach used to compute the spectrum could possibly explain the relatively high peaked behavior in the time series and variance density spectrum comparisons. At the same time, this does not explain the inadequacy of the numerical model to capture the smaller wave amplitudes.

These results indicate that there is adequate resolution of the offshore wave conditions in the numerical wave tank. As a result, further comparison with the experiments can be carried out. In the following sub-section, the results for `OpenFOAM` and OceanWave3D will be discussed to evaluate the coupling between the two models.

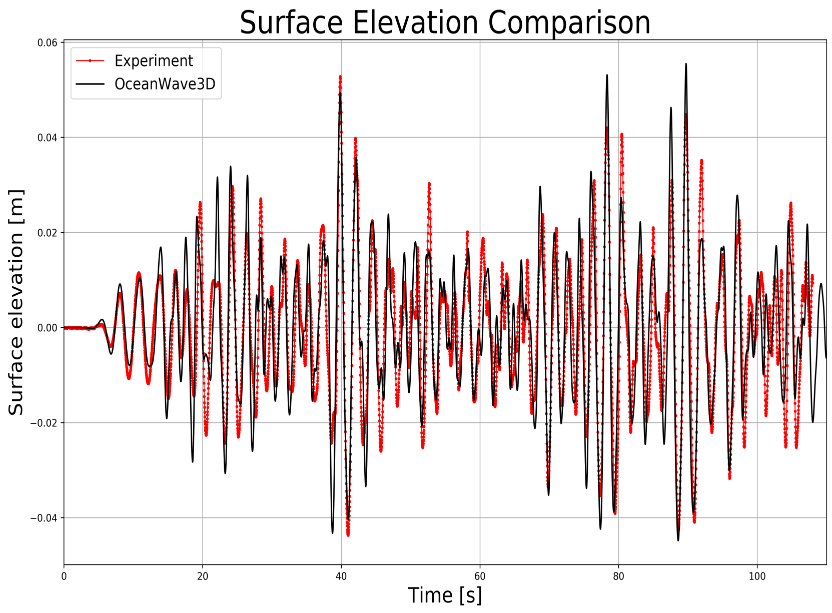


(a) Time Series Comparison

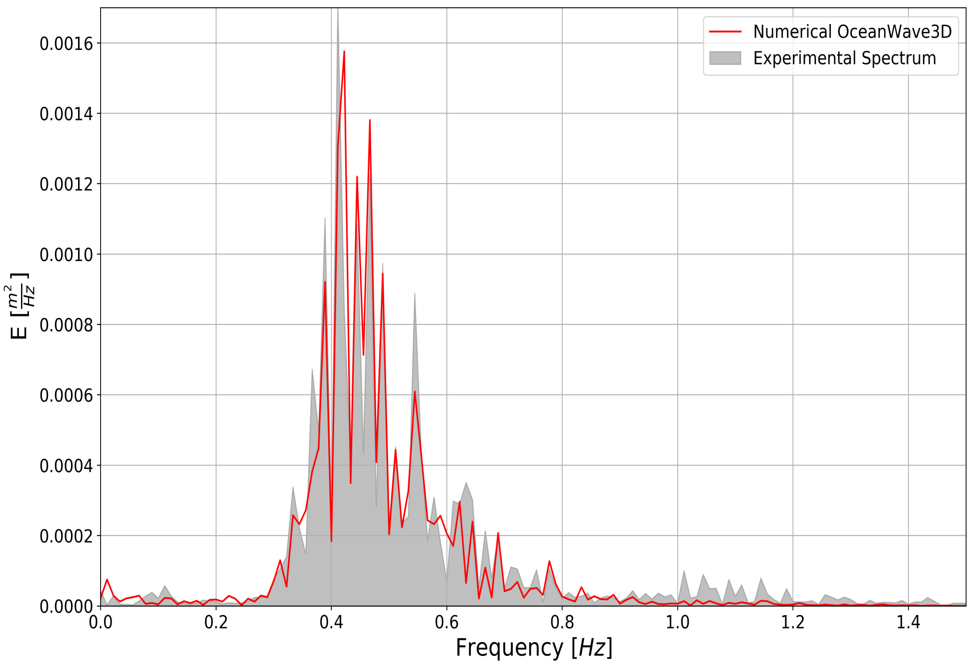


(b) Variance Spectral Density Comparison

Figure 5.15: Time and Frequency domain comparison of the surface elevation at WG2 ($x \sim 3.415$ m).

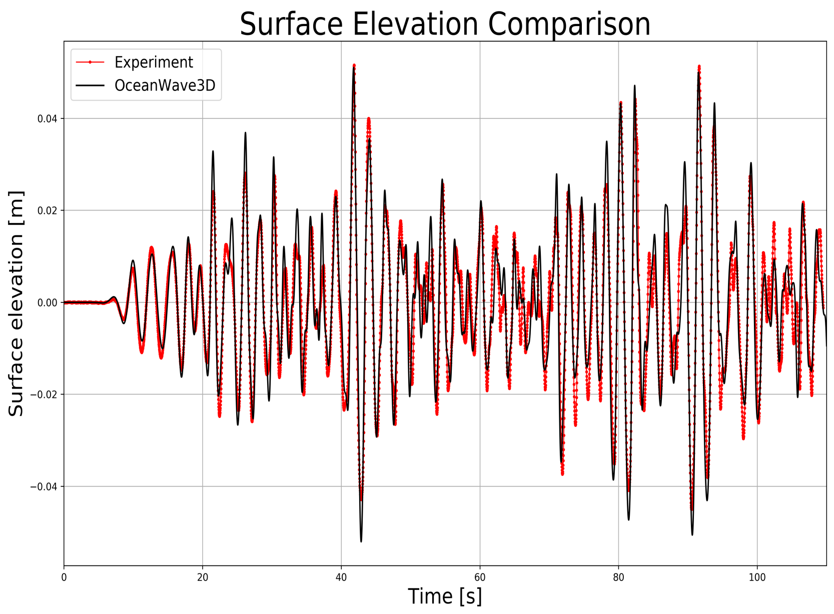


(a) Time Series Comparison

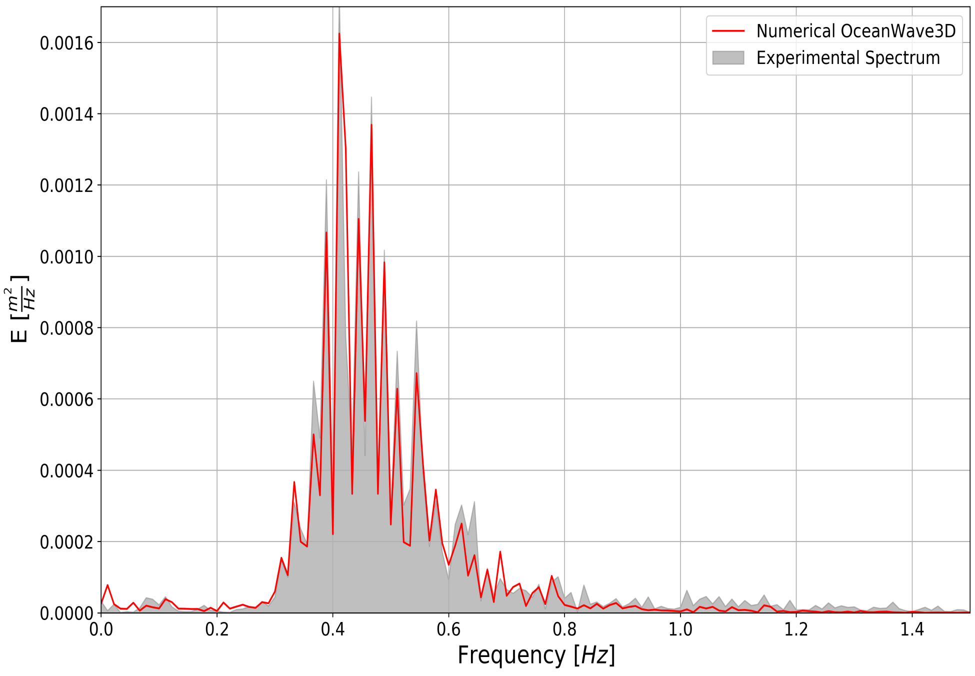


(b) Variance Spectral Density Comparison

Figure 5.16: Time and Frequency domain comparison of the surface elevation at WG5 ($x \sim 9$ m).

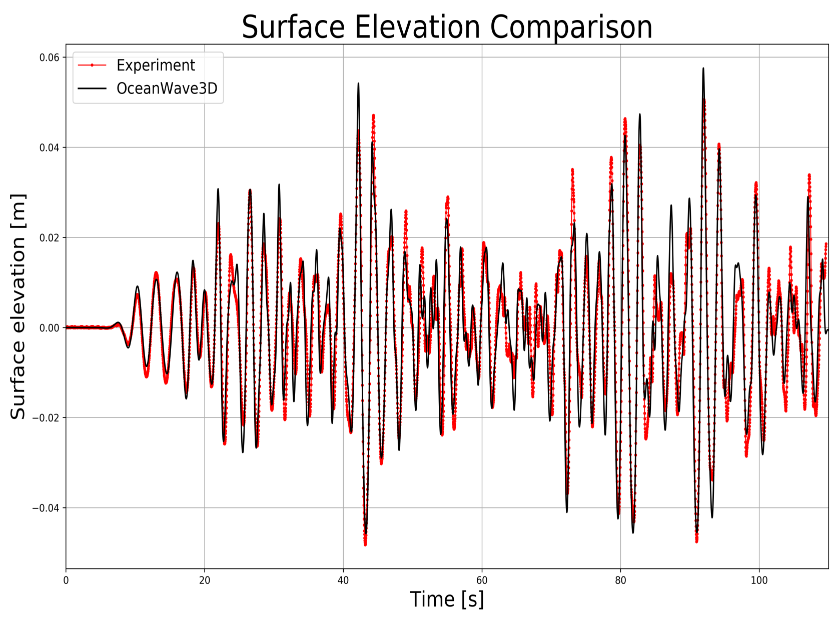


(a) Time Series Comparison

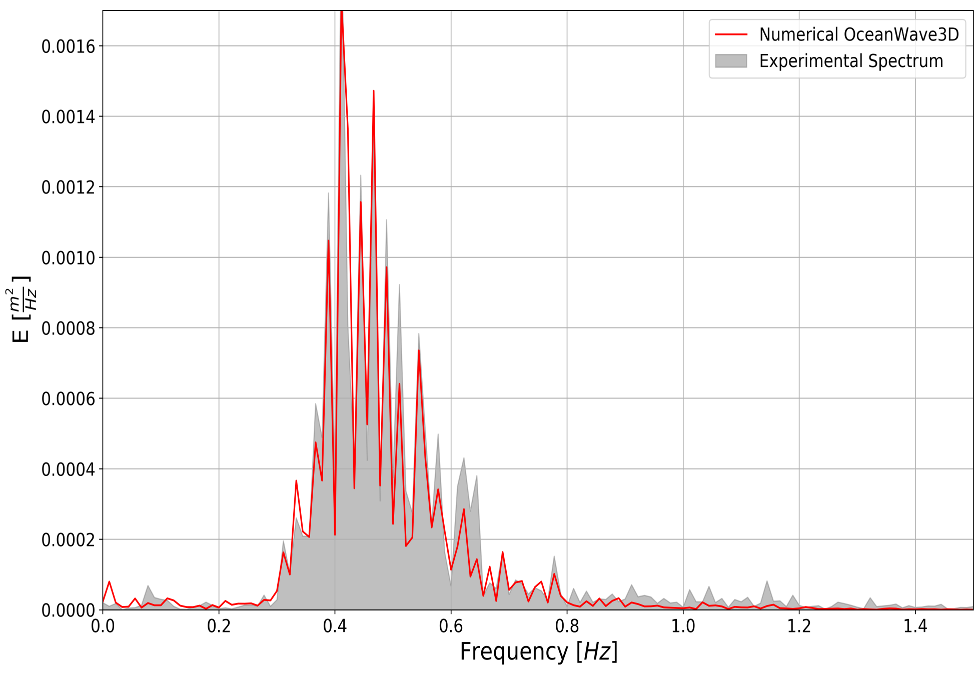


(b) Variance Spectral Density Comparison

Figure 5.17: Time and Frequency domain comparison of the surface elevation at WG2 ($x \sim$ 13.94 m).



(a) Time Series Comparison



(b) Variance Spectral Density Comparison

Figure 5.18: Time and Frequency domain comparison of the surface elevation at WG2 ($x \sim$ 14.84 m).

5.2.2 Surface elevation comparison of OW3D and OpenFOAM

As seen in Figure 5.19, the surface elevation comparison of the coupled numerical models shows good agreement at $x \sim 35.54$ m in the domain. Although this wave gauge lies within the relaxation zone, the lack data availability and the wave breaking occurring at around $x \sim 41$ m makes it difficult to directly compare the OW3D and `OpenFOAM` results in the rest of the computational domain. The minor differences between the two models can be attributed to the relaxation zone and the coarse grid resolution of the `OpenFOAM` domain. Despite these issues, the comparison between the two models is sufficient.

Since the offshore surface elevation comparison shows good comparison between the numerical results and the experimental results in addition to the wave gauge within the `OpenFOAM` domain, it can be safely concluded that the wave conditions incident in the `OpenFOAM` domain conform to the ones assigned at the wave paddle. As a result, this concludes that the coupling between the two models works as expected. This motivates further analysis of the overtopping results using the numerical results.

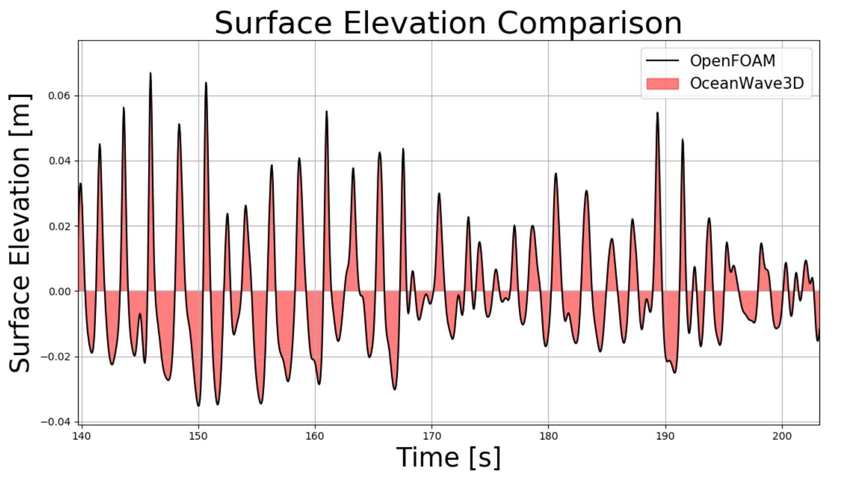


Figure 5.19: Surface elevation comparison of OceanWave3D and `OpenFOAM` at a wave gauge located at $x \sim 35.54$ m.

5.2.3 Comparison of surf-zone surface elevations

In order to understand the distribution of the surface elevation across the surf zone, description of the $\eta_{s,max}$ and $\overline{\eta}$ was carried out. To compute the $\eta_{s,max}$ a zero crossing analysis was used to distinguish individual waves[1]. After each wave was separated, the maximum surface elevation within each wave was identified as the $\eta_{max}[i]$ for the i^{th} wave. Using a collection of such $\eta_{max}[i]$ the highest $1/3^{rd}$ η_{max} were averaged to arrive at the estimation of $\eta_{s,max}$. A sample plot used in the analysis can be seen in the Figure 5.20 below.

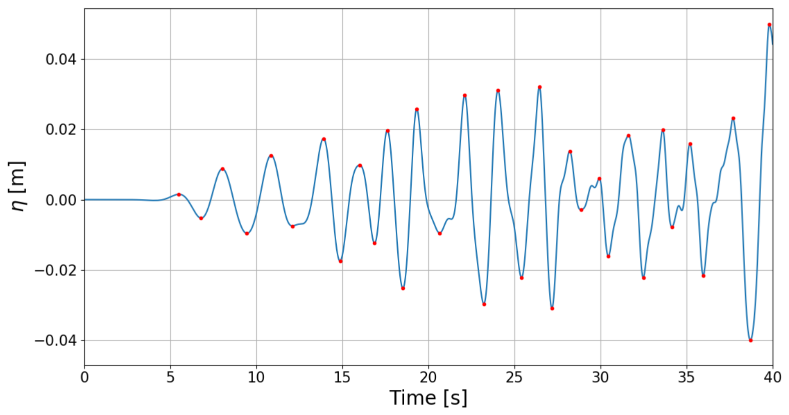


Figure 5.20: Sample minimum-maximum analysis used in this investigation. The red dots represent the maximum and minimum surface elevations as detected by the algorithm used, while the blue curve represents the original η time series.

[1]The python code for this has been included in Appendix D

Using this analysis approach the distribution of $\eta_{s,max}$ and $\overline{\eta}$ has been presented in Figure 5.21. As opposed to the Ting and Kirby [1994] results, there seems to be no phase lag between the wave breaking location (X_b) and the initiation of the wave induced set-up. This gives a preliminary indication that wave breaking transforms most of the kinetic energy into turbulence, while only small amount of momentum is retained as the roller progresses and dissipates the rest of the energy. This is true for the bigger waves in the wave group, while the smaller waves almost all the time convert the kinetic energy into turbulence. The larger waves in the domain tend to break earlier while the smaller waves break later (deeper within the surf-zone) as a result of wave steepening. On average the waves break at around $x \sim 40.79m$ as predicted by the numerical model. The schematized wave flume in the figure has been presented for reference purposes only and the still water depth at $x \sim 34m$ is $0.2m$.

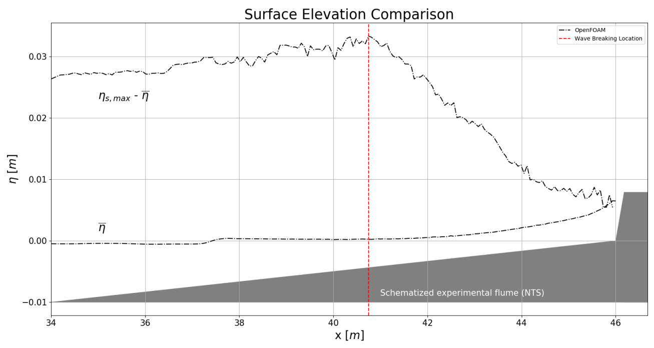


Figure 5.21: Distribution of maximum surface elevation and mean surface elevation across the surf-zone using the `waveFlow` solver. The extend of the plot corresponds to the domain within `OpenFOAM` (See Figure 4.9b).

5.2.4 Overtopping comparison

This section presents a comparison of the wave overtopping for three datasets viz., *experiment*, `waveFoam`, and `waveFlow` solver's respectively. As seen in Figure B.3, a similar over-prediction in the overtopping discharge is observed. This is clearly seen in the cumulative overtopping comparison presented in Figure 5.23. The coarse grid results in scalar flux advection in both the solvers even when there is not wave incoming which results in an overtopping event. This problem is especially critical for the `waveFoam` solver since the interface is relatively more diffused. As a result, this results in a positive overtopping signal in the time period ($t < 45$ s) where the experimental results show no overtopping occurring. However, the volume that is overtopped in `waveFoam` is significantly higher in comparison to that in `waveFlow`.

The first grid cell in the coarse grid setup is relatively large in comparison to the approach used in Section 4.1.1. This setup was used to check the overall performance of the numerical wave tank setup. However, the first grid cell is where most of the overtopping flux is concentrated since the overtopping observed in the experimental setup is not violent in comparison to that observed in Section 3.3.2. The localization of the flux of fluid around the first grid cell in the `waveFlow` solver can be seen in the series of snapshots in Figure 5.24. The figure depicts that the flux passing over the dike is concentrated within a single cell as it traverses over the dike and is registered by the overtopping sampling plane as a positive instantaneous overtopping discharge. This leads to a peaked signal in the overtopping time series for the `waveFlow` solver. However, the flux passing through the overtopping plane in the `waveFoam` solver is diffused over three grid cells in the vertical direction and more than a few grid cells in the stream-wise direction, thus portraying a more continuous and long lasting overtopping event.

As seen in Figure B.3, the overtopping signal for `waveFoam` solver indicates a positive overtopping discharge throughout the time series. This results in a positive overtopping volume for longer time in comparison to `waveFlow` resulting in a grossly over-predicted cumulative overtopping volumes.

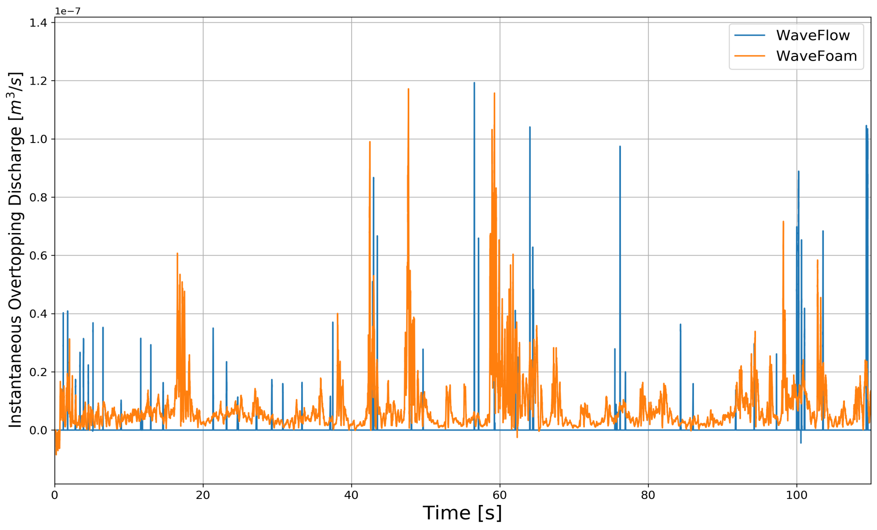


Figure 5.22: Instantaneous overtopping discharge time series comparison of the two solvers under consideration.

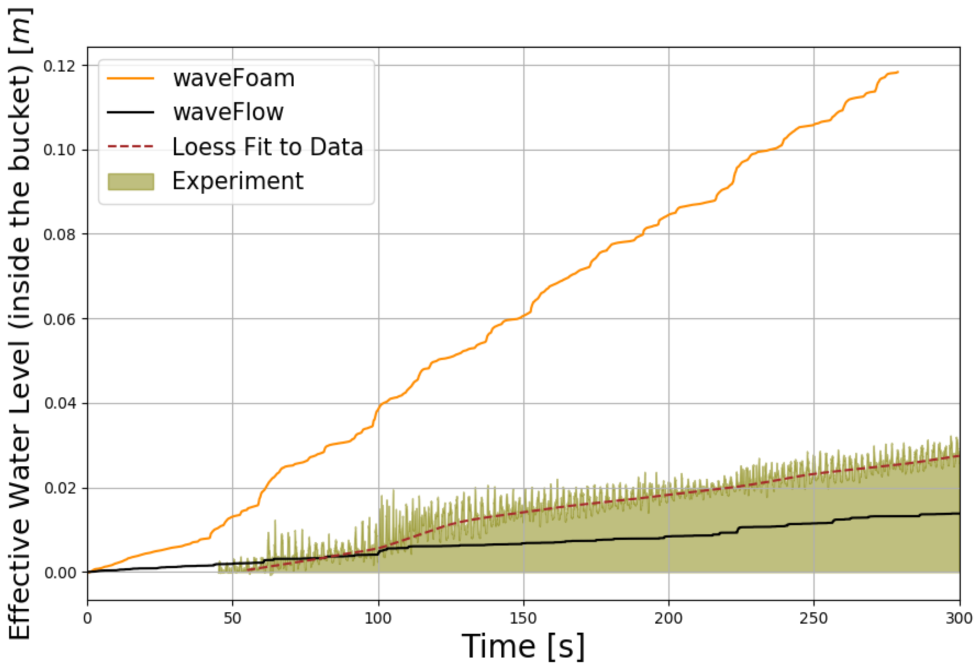
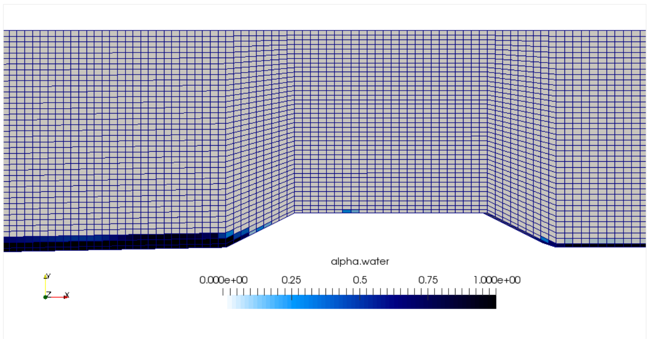
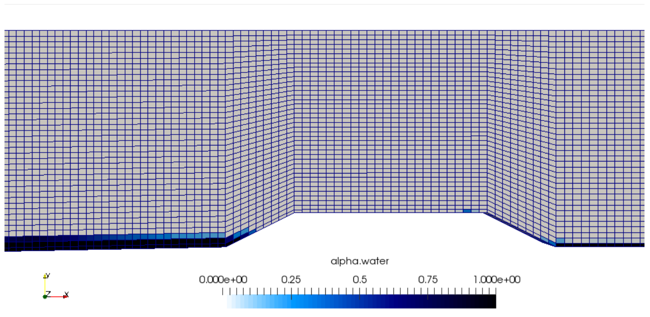


Figure 5.23: Cumulative overtopped volume time series comparison of the two solvers under consideration. The results have also been compared to the experimental results. The shaded region represents the actual time series of the experimental results. The Loess fit line is basically a local regression fit to the oscillating experimental signal.



(a) T = 66 s



(b) T = 66.5 s

Figure 5.24: Overtopping behavior in the numerical wave tank for the `waveFlow` solver

Given that most of the wave energy is lost due to turbulent wave breaking in the surf zone Altomare et al. [2016], very little momentum persists leading to run-up followed by an overtopping event. As a result, the overtopping observed in the numerical wave tank and the experimental are small in magnitude. However, despite the coarse grid effects, `waveFlow` seems to be predicting overtopping volumes closer to the experimental findings, thus confirming the underlying hypothesis in this study. This accuracy however, comes at a 10-15 % additional computational cost.

The results presented above correspond to the coarse mesh results with a spatial resolution of about $dx \sim 0.025$ m. Although the numerical results seem to be in agreement to the experimental results, the lack of feasibility for a sensitivity analysis for grid effects can possibly undermine the results. Consequently a finer grid was considered in order to analyze the overtopping trend. However, no sensitivity study was carried out. The grid resolution in this case was $dx \sim 0.0068m$. This results in $\sim$ 7 grid points for the smallest offshore wave height ($\sim 0.0483m$) and $\sim$ 17 grid points for the largest offshore incident wave height ($\sim 0.121m$). The surface elevation comparison for both the models are similar to what was presented in the preceding text. The only differences observed were that in the overtopping time series. As seen in Figure 5.25, the overtopping results follow the experimental results for the `waveFlow` solver. The over-prediction trend for `waveFoam` is consistent in all the studies carried out in this thesis work[2].

[2]XBeach results were not a part of the work done in this thesis

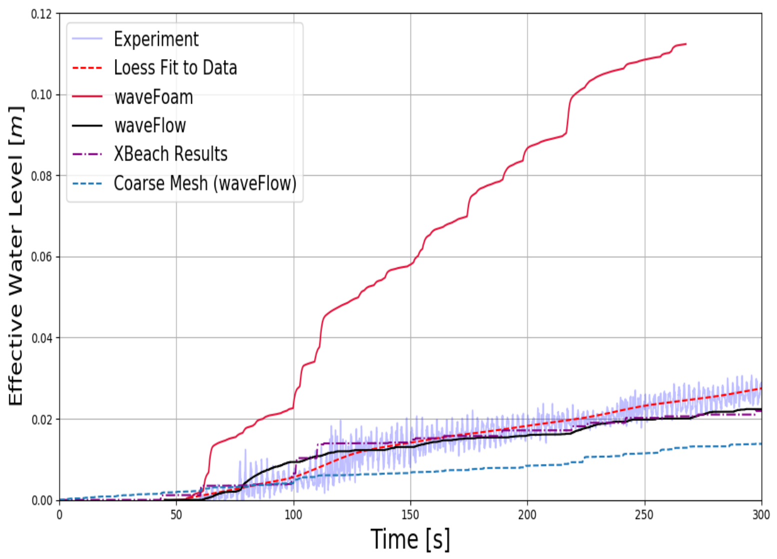


Figure 5.25: Cumulative overtopped volume time series comparison of the two solvers under consideration

Considering the previous argument of constant positive overtopping in `waveFoam`, a threshold of $Q(t) = 10^{-6}\ m^3/s$ was set to understand the isolate the performance of `waveFoam`. The clipped overtopping signal can be seen in Figure 5.26. Despite the filtering of small overtopping events, the solvers seems to be over-predicting overtopping volume for almost every individual overtopping event. This overprediction can be clearly seen in Figure 5.27. This provides substantial proof regarding the overtopping behavior for the two solvers.

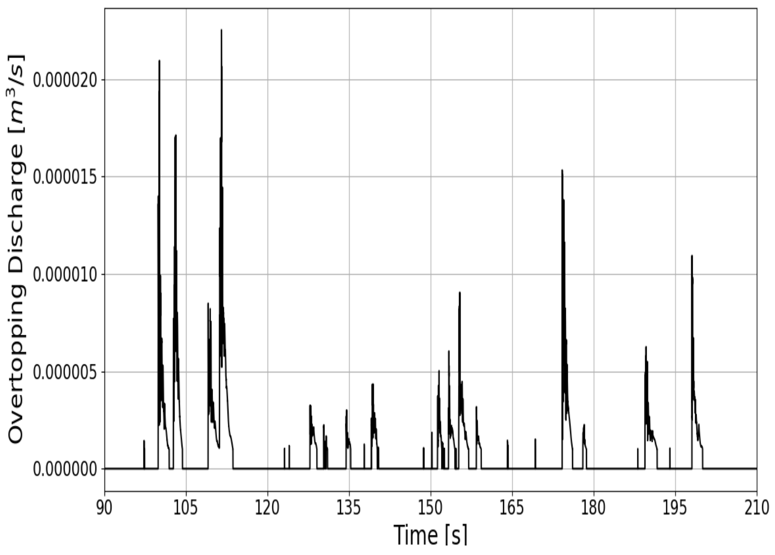


Figure 5.26: Overtopping discharge filtered time series.

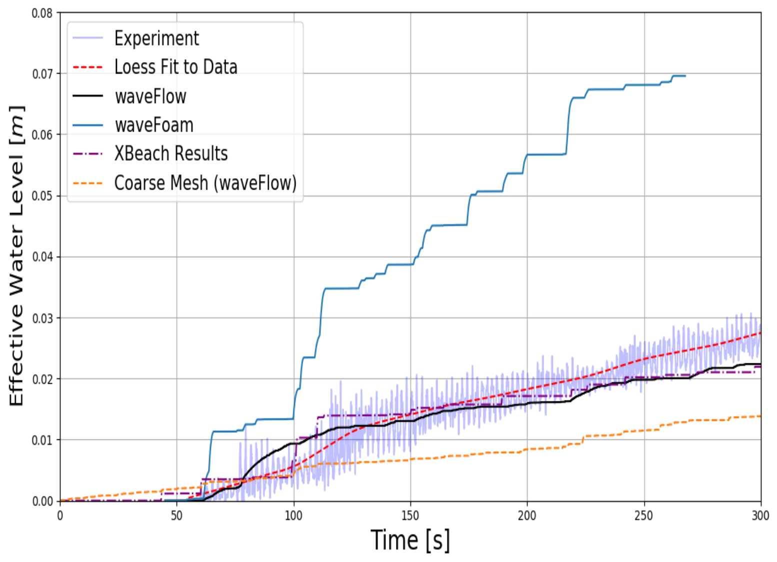


Figure 5.27: Cumulative overtopped volume for the filtered version of the overtopping discharge.

Overtopping results without the overpredicting `waveFoam` solver can be seen in Figure 5.28. `OpenFOAM` solvers seem to be predicting the small as well as large overtopping volumes quite reasonably. However, since the grid effects are not quantified it is expected that the results will improve upon further refinement, but this was not possible in this investigation. This extensive comparison confirms the quantitative and qualitative differences between the two interface capture methods used in this study. Within the scope of the grid and computing environment discussed in this investigation, `waveFlow` solvers seems to produce the closest results for surface elevation and overtopping simultaneously, while `waveFoam` seems to overpredict the overtopping volumes by a large factor.

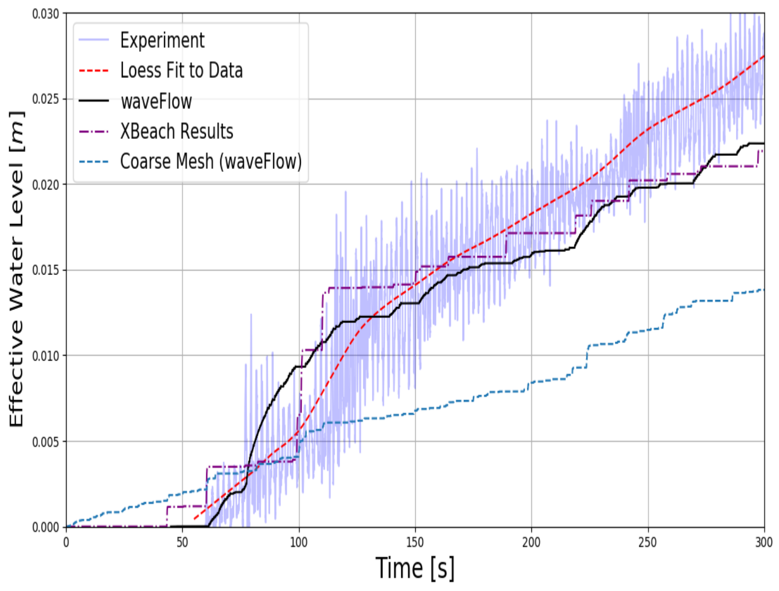


Figure 5.28: Cumulative overtopped volume comparison against different databases used in this study, without the `waveFoam` solver.

6 CONCLUSIONS AND RECOMMENDATIONS

OUTLINE

This chapter provides initially the conclusions for the research questions investigated during this thesis work. It is followed by some general recommendations about using the new solver along with some practical usage recommendation.

CONCLUSIONS

What is the practical feasibility of advanced CFD based models in comparison to simpler models like phase averaged and phase-resolving models, in complex surf zones and swash zones?, and which physical processes in the surf zone affect wave overtopping on the slopes of coastal structures like dikes?

6.1 PRACTICALITY OF COMPUTATIONAL FLUID DYNAMICS

The practicality of the CFD models as used in this particular investigation can be looked at in two different perspectives as listed below and will be discussed in the following sub-sections.

- Predicting mean properties for long time durations
- Predicting detailed hydrodynamics for short time durations

6.1.1 Long time durations

The very first approach would be to obtain mean properties viz., mean η distribution across the shore, mean overtopping volume, run-up, and possibly other hydrodynamics in the surf zone using an advanced model. The present coarse grid solutions with the new development (`waveFlow`) show promising results, especially for surface elevation and overtopping measurements. Despite the coarse resolution, the numerical results using the new solver seem to predict better overtopping results in comparison to its predecessor solver. However, the feasibility of using such a detailed model is critically limited to a few key parameters as listed below.

- Optimizing computational resources for a given HPC system.
- Necessity for large time durations (to replicate complete sea state conditions).

The first point in the list above has a fundamental limitation on how much speed-up can be achieved for a given piece of code on a specified computing environment. In this particular case, two computing clusters were used to test the feasibility of the C++ code used. Most of the computational effort was spent in replicating the rather long time series than solving intricate physics, say Large Eddy Simulation or Direct Numerical simulations. Given that a RANS turbulence closure was used, the grid resolution requirement was relatively low. As a result, distributing the computational geometry over 'n' number of computing processors only speed-up the simulation when $n \sim 15 - 16$. Any additional computing processors did not add computational value. Consequently, the major challenge in this study was replicating the long time domain. As a result, optimizing the computation itself was not possible. However, as discussed in section 2.4.2, shortening the length of the time domain and analyzing the key hydrodynamic properties for design seems to be valid approach for wave overtopping applications. Although, this has theoretical limitations since it does not replicate all the crucial frequencies in the sea state spectrum. This also details the second point in the list above.

To recapitulate, the current feasibility of CFD models for long term wave applications seems to be in the primitive stages for long duration wave model applications. This is despite using a coupled modeling approach like the one used in this case. Given that no speed-up tests were done to understand the scalability of the computing cluster at TU Delft, one of the recommendations would be to investigate the behavior of the locally compiled code on the computing cluster. As seen in Figure 6.1, a similar scalability test will greatly help in estimating the computational optimum for the given computing environment.

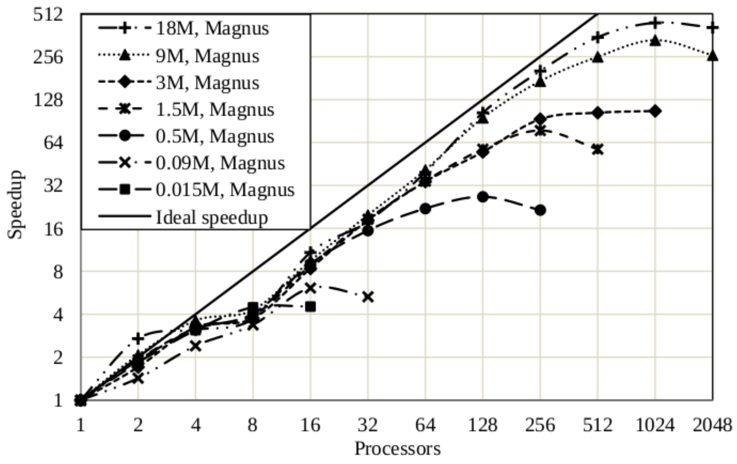


Figure 6.1: Scalability on the 'Magnus' computing cluster at PAWSEY, Perth. Adapted from [Tong, 2014].

Although the numerical results using `OpenFOAM` demand larger computational time in comparison to the simpler models, the overtopping results using XBeach fail to resolve small overtopping events but preserve the general overtopping trend. Consequently, using a CFD model capable of simulating wave breaking can prove effective in the end in order to predict small and large overtopping events which otherwise cannot be captured using a simpler model.

6.1.2 Detailed Hydrodynamics

As already detailed in Jacobsen et al. [2012], Larsen and Fuhrman [2018], and Brown et al. [2016] to list a few, the use of `OpenFOAM` yields sufficiently detailed results in terms of surface elevation, turbulence levels, velocity profiles, and possibly other hydrodynamics not mentioned in this list. Instead of using the CFD model as a preliminary design tool to obtain mean hydrodynamic quantities, looking at the most critical design events to gain better insights into the system behavior seems to be an effective approach to utilize such a numerical model.

Given that the numerical wave tank is capable of providing detailed predictions for a large number of hydrodynamic parameters, the feasibility of such numerical model lies in detailed investigation of a limited time domain. However, having said this the computational time required for a coarse grid simulation to run about 300 s of real time, an estimated 6 days were required. In comparison to a wide variety of other simpler models (SWAN, SWASH, XBeach etc.) the computational time is exponentially large. However, given the amount of details that can be obtained the feasibility of such a CFD model is based on the required hydrodynamic parameters. To conclude, using CFD models for long time durations boils down to a balance between user requirement and computational feasibility. For detailed results, a CFD model can provide relatively sufficient resolution in the hydrodynamics, while other simpler models can be used to predict the mean hydrodynamics. This answers the first part of the research question under investigation.

6.2 WAVE OVERTOPPING USING *waveflow*

As detailed in the numerical model results discussed in Section 5.2, wave breaking was observed to be a key criterion in the resulting wave overtopping. Given that most of the waves break quite some distance away from the toe of the dike, the roller propagating as a result of the wave breaking is seen to have reduced onshore momentum due to successive splash up cycles. The spectral analysis of the wave signal at the toe of the dike (See Figure 6.2) shows that a substantial amount of energy exists in the higher frequency side after the waves have broken while most of the wave energy has been transfered to lower frequency post breaking. `OpenFOAM` fails to capture the low frequency peak but shows the energy on the higher side of the spectrum. OceanWave3D on the other hand is able to capture the peak frequency but fails to capture the high frequency tail of the spectrum. This shows the stark differences between the two model types.

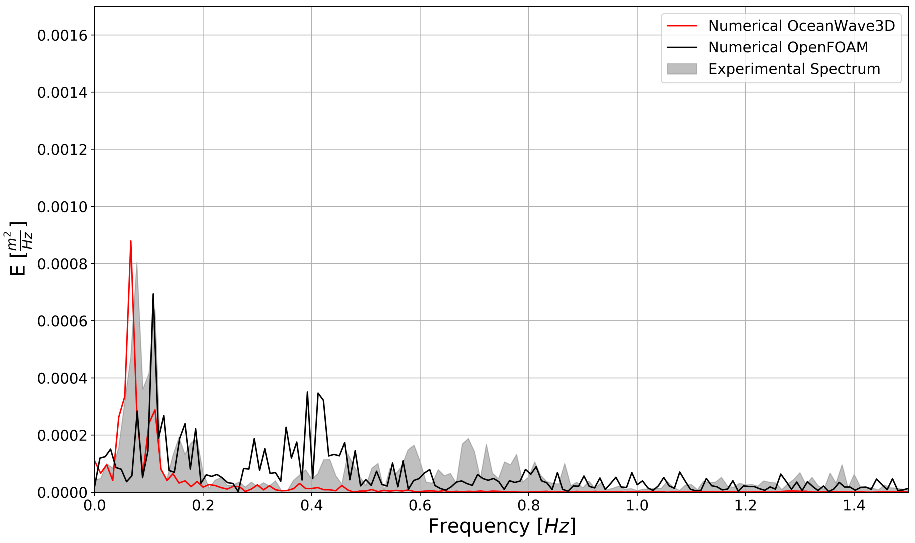


Figure 6.2: Variance density spectrum at the toe of the dike ($x \sim 45.79$ m)

Note: No frequency analysis done for the low frequency component contributing to overtopping.

The new development within the `OpenFOAM` framework seems to be providing a relatively realistic estimate in comparison to its predecessor solver. However, the results correspond to a coarse grid simulation thus limiting the reliability of the results as presented in their present form. Despite the coarse grid resolution, the wave overtopping results using the new solver are relatively in better agreement with the experimental results, while the previous solvers seems to be grossly overpredicting the overtopping volumes for the same grid size. Further investigations using an intermediate grid size reveal that `OpenFOAM` consistently over-predicts the overtopping volumes by a factor of 3-5 in comparison to the experimental results. The new solver seems to provide a reasonable resolution for wave overtopping in comparison to the experiment. It is expected that further refinement and isolating the effect of grid on the results can improve the overtopping predictions. Thus it can be safely concluded as evidenced in the preceding text that new development within the OpenFOAM framework provides improved and relatively more accurate overtopping results in comparison to the previous solvers.

Sub-Questions

- *What hydrodynamics of the surf-zone can be obtained with an advanced model employing a RANS based approach?*

  The turbulence model stabilization coefficients as discussed in Larsen and Fuhrman [2018] contribute to the variation in the wave breaking behavior. Using the current λ_2 value results in a better agreement for the plunging breaker case where larger TKE is expected due to the violent plunging jet generated. As outlined by Larsen and Fuhrman [2018] and Brown et al. [2016], getting bounded and accurate TKE and velocity structure within the surf zone seems to be still an open problem. However, using the developments made by Larsen and Fuhrman [2018] seem to yield sufficiently bounded TKE values unlike the previous turbulence closures which are polluted with erroneously large TKE value in the pre-breaking region. Despite the over-prediction of TKE deep within the surf zone, the mean wave height distribution was well captured by the numerical model for both the spilling and plunging breaker. The velocity and TKE distribution barring few over-predictions also seem to provide a reasonable estimate in comparison to the experimental results.

  Significant improvements were obtained in the plunging breaker case, these improvements can be attributed to the changed stabilization coefficients. Although a new interface capture method was used, the differences observed in the surface elevation plots especially for the plunging case can be attributed to the changed λ_2 parameter in the turbulence model. This allowed for larger turbulence levels to be developed within the fluid domain where a potential flow region exists. Substantial evidence that no significant changes were observed between the two solvers for the same turbulence model can be seen in Figure 6.3. The figure presents the same comparison as seen in Figure 5.2 but in this case for `waveFoam` solver. All the other settings for this simulation were identical to that used for `waveFlow` solver. This clearly shows that there is no substantial difference between the results obtained by new solver. Resolving wave breaking results in better surface elevation and wave setup predictions using the stabilized RANS closure. However, the results obtained are quite sensitive to the stabilization parameters used to simulate turbulence. This was already shown in Larsen and Fuhrman [2018] and has been replicated in this study with minor improvements for the plunging breaker case.

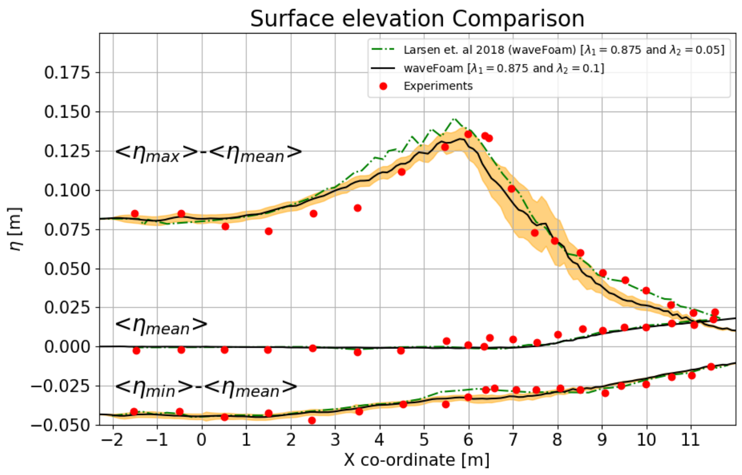


Figure 6.3: Surface elevation comparison for `waveFoam` solver with identical turbulence model settings as used for `waveFlow` solver.

- *How is the turbulent kinetic energy distributed for spilling and plunging wave breaker using a RANS based approach?*

  The TKE in the numerical model is over-predicted as one moves deeper within the surf-zone. Closer to the wave breaking point, the TKE profiles seem to follow the experimental findings adequately. In both the spilling and the plunging case, the TKE is grossly over-predicted for the two most onshore wave gauge's as seen in Figure 6.4.

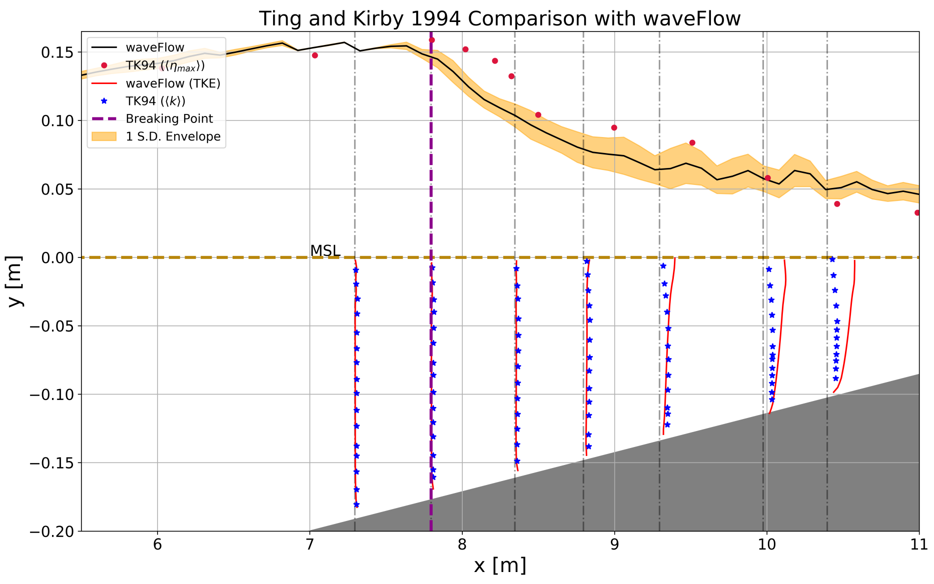


Figure 6.4: Turbulent Kinetic Energy profile comparison for plunging breaker case.

  The details have been discussed in Section 5.1.2.

- *What additional improvement can be observed in the prediction of wave overtopping over coastal dikes using a RANS based approach in comparison to simpler models?*

  Significant improvements can be seen in the newly integrated `waveFlow` solver in comparison to the previous solver. The new solver seems to be predicting more realistic overtopping results in comparison to the previous solver. These results are more in line with the experimental results as seen in Figure 5.23. Despite the coarse mesh, the results for overtopping volume are similar to the overtopping results predicted by using XBeach (See Figure 5.28). The coarse mesh consists of about 7 grid points for the smallest wave height incident in the domain. Using a finer grid with about 17 grid points for the smallest wave height incident in the domain results is satisfactory overtopping predictions using the `waveFlow` solver. It is expected that the results using a finer mesh will provide conclusive proof after estimating the effect of grid on the overtopping results. Thus demanding a sensitivity analysis for the case, due to time constraint this was not feasible in this investigation.

A practical outlook on the conservative prediction of `waveFlow` could result in under-designed coastal defenses. As a result, carrying out a sensitivity analysis for the effect of grid on the results is recommended. This will provide an informed conclusion about the performance of the numerical model.

- *What physical processes contribute to wave overtopping and how well are they replicated in the numerical model?*

  Wave breaking seems to be one of the prominent processes affecting wave run-up and overtopping. As discussed in the previous chapter, wave breaking results in a propagating bore and the strength of this bore is characterized by the type of wave breaking. In general, plunging breakers create splash up cycles resulting in a stronger bore and hence larger overtopping events for the same geometry. This splash up cycle results in a phase lag between the wave induced setup and wave breaking location which was also observed and investigated in Jacobsen et al. [2014]. Large momentum retention in the post wave breaking region results in a higher run-up since the roller can propagate further into the surf-zone.

  The numerical model sufficiently resolves wave breaking and the associated characteristics using a simple RANS closure. The wave overtopping resolution for large and small overtopping events is satisfactorily captured by `waveFlow`. This is in contrast to what was observed for the XBeach results, which failed to capture the small overtopping events. This overtopping behavior can be seen in Figure 6.5. In conclusion the numerical model using *waveFlow* seems to provide accurate representation of the surface elevation, wave breaking behavior, and wave overtopping for the investigations carried out in this particular study.

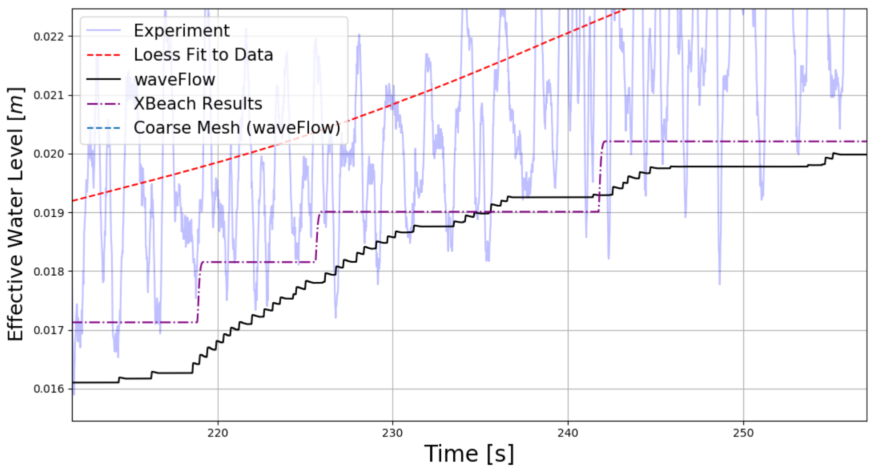


Figure 6.5: Overtopping signal comparison of small overtopping events as observed by XBeach and `waveFlow` solver.

RECOMMENDATIONS

This section details some practical usage recommendations as experienced during this investigation.

6.2.1 Ting and Kirby [1994] Validation Test

- The results obtained in this test case were found to be sensitive to the grid aspect ratio, so much so that a mean aspect ratio of about 1.3 in the domain results in premature wave breaking. This was already mentioned in Jacobsen et al. [2012] and has been reaffirmed in the current investigation. The results using an aspect ratio of 1.3 can be seen in Figure **??**. In order to correctly capture wave breaking location and height, an aspect ratio of 1 should be maintained in the numerical domain.

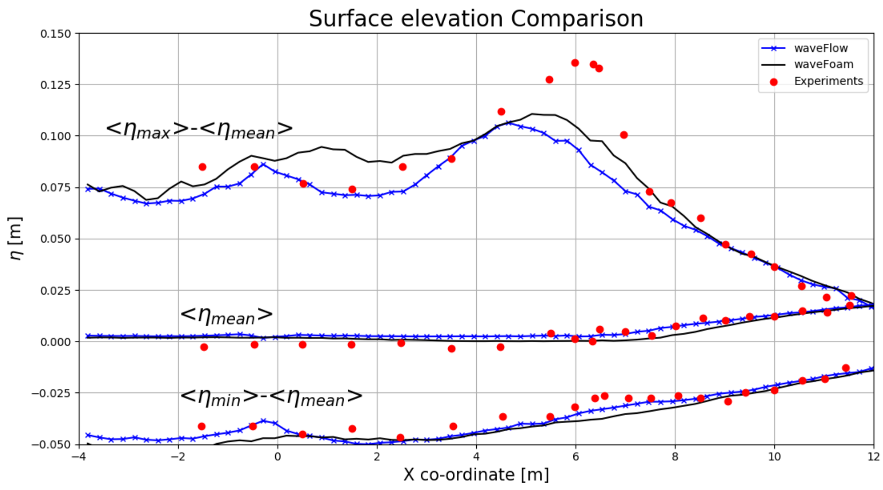


Figure 6.6: Premature breaking for both the solvers using poor quality mesh.

- It was observed that the wave breaking behavior is sensitive to the choice of stabilization parameters used in the RANS closure developed by Larsen and Fuhrman [2018]. Consequently, understanding the effect of the choice of various λ parameters can result in different wave breaking behavior. This was already outlined and stressed in Larsen and Fuhrman [2018]. In addition, the current investigation showed substantial improvement in the plunging breaker behavior by using a different λ parameter combination.

- The sampling rate for TKE and undertow measurement for the spilling case should be kept to a high number since the wave period is small. A small sampling frequency results in limited data for a given wave period and thus no sensible averaging can be obtained. A small sampling frequency results in an over-predicted and potentially biased profile as seen in Figure 6.7.

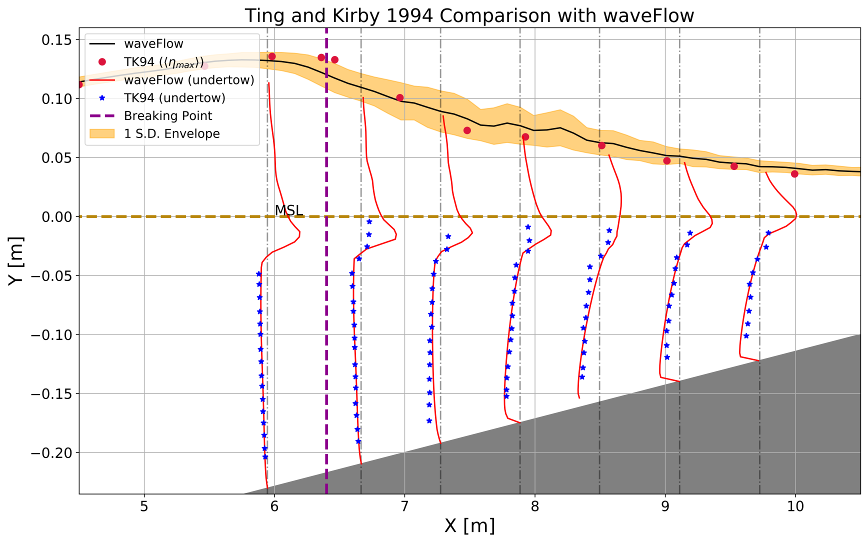


Figure 6.7: Incorrect velocity profile using small sampling frequency ($f \sim 10Hz$) for sampling the data.

6.2.2 Flanders Experimental Test

- Optimizing the computational time can provide significant speedup in the simulations. As a result, it is advised to carry out a small scaling test for the given setup. It was observed that in general for a mesh count of 0.1 Million cells, a total of 16 processors results in the most optimal configuration. However, this number is also dependent on the hardware used on the computing cluster.

- The write precision in OpenFOAM required for overtopping sampling should be set to atleast 8 decimal places. A smaller write precision results in constant time steps for certain parts of the simulations where the Δt reaches extremely small values. Since this Δt is used in the computation of the overtopping volume, it may erroneously result in an overtopping event with $\sim 0[m^3]$ overtopping volume since the $\Delta t \sim 0[s]$. As a result, increasing the write precision within the *controlDict* can solve this issue.

- Overtopping measurements were found to be sensitive to the grid cell layers close to the bottom wall. This was especially true for small overtopping events. Using an extremely fine mesh near the bottom wall region resulted in flux advection even when there was no overtopping observed in the experiments. As a result, having a sufficiently fine mesh so as to stay within the $y^+ < 30$ region and not critically affect the overtopping results had to undergo a trial and error iteration.

- Additional investigations are proposed in order to investigate the differences between `waveFlow` and `waveFoam` solvers. As discussed above, the overtopping results improve significantly using a grid with ~ 17 grid cells for the smallest wave height incident in the domain. Isolating the effect of grid can provide a better estimate for the effect grid size on the overtopping predictions using both the solvers. Due to the limited scope of this investigation, it was not possible to simulate the entire time series involving 1000 waves. As a result, a detailed investigation employing 1000 waves is recommended to further establish the efficacy of `OpenFOAM` for overtopping computations.

BIBLIOGRAPHY

Altomare, C., Suzuki, T., Chen, X., Verwaest, T., and Kortenhaus, A. (2016). Wave overtopping of sea dikes with very shallow foreshores. *Coastal Engineering*, 116:236 – 257.

Arcilla, S. and Lemos, C. M. (1990). Surf-zone hydrodynamics. *Centro Internacional de Metodos Numericos de Ingenieria*, page 310.

Argyropoulos, C. and Markatos, N. (2015). Recent advances on the numerical modelling of turbulent flows. *Applied Mathematical Modelling*, 39(2):693 – 732.

Battjes, J. (1974). Surf similarity. *Proceedings of the 14th Coastal Engineering Conference*, pages 466–480.

Battjes, J. (1988). Surf-zone dynamics. *Annual Review of Fluid Mechanics*, 20:257–293.

Beij, S. and Battjes, J. (1993). Experimental investigation of wave propagation over a bar. *Coastal Engineering, Elsevier*, 19:151–162.

Benjamin, T. (1967). Instability of periodic wavetrains in nonlinear dispersive systems. *Proceedings Royal Society*, 299:59–75.

Benjamin, T. B. and Feir, J. E. (1967). The disintegration of wave trains on deep water part 1. theory. *Journal of Fluid Mechanics*, 27(3):417–430.

Bosboom, J. and Stive, M. (2011). *Coastal Dynamics : Part 1 (version 2011-0.2)*, volume Part 1 (version 2011-0.2). VSSD. Lecture Notes CT4305.

Brocchini, M. and Dodd, N. (2008). Nonlinear shallow water equation modeling for coastal engineering. *Journal of Waterway Port Coastal and Ocean Engineering-asce - J Waterways Port Coast OC-ASCE*, 134:104–120.

Brown, S. A., Greaves, D. M., Magar, V., and Conley, D. C. (2016). Evaluation of turbulence closure models under spilling and plunging breakers in the surf zone. *Coastal Engineering, Elsevier*, 114:177–193.

C Yuen, H. and M Lake, B. (1980). Instabilities of waves on deep water. *Annual Review of Fluid Mechanics*, 12:303–334.

Carrier, G. and Greenspan, H. (1958). Water waves of finite amplitude on a sloping beach. *Journal of Fluid Mechanics*, 4:97–107.

Chalikov, D. (2007). Numerical simulation of the Benjamin-Feir instability and its consequences. *Physics of Fluids*, 19(1):016602.

Christensen, E. and Deigaard, R. (2001). Large eddy simulation of breaking waves. *Coastal Engineering*, 42:53–86.

Coastal Engineering Research Council (1984). Shore Protection Manual Vol I and II. *Department of the Army, US Army Corps of Engineers, Washington, DC, 20314*, 4.

Devolder, B., Troch, P., and Rauwoens, P. (2018). Performance of a buoyancy-modified $k-\omega$ and $k-\omega$ SST turbulence model for simulating wave breaking under regular waves using OpenFOAM®. *Coastal Engineering, Elsevier*, 138:49–65.

Engsig-Karup, A., Bingham, H., and Lindberg, O. (2009). An efficient flexible-order model for 3D nonlinear water waves. *Journal of Computational Physics*, 228:2100–2118.

EurOtop, Van der Meer, J., Allsop, N., Bruce, T., De Rouck, J., Kortenhaus, A., Pullen, T., Schuttrumpf, H., Troch, P., and Zanuttigh, B. (2018). Eurotop, manual on wave overtopping of sea defences and related structures. an overtopping manual largely based on european research, but for worldwide application. www.overtopping-manual.com.

Fuhrman, D. R., Madsen, P. A., and Bingham, H. B. (2006). Numerical simulation of lowest-order short-crested wave instabilities. *Journal of Fluid Mechanics*, 563:415–441.

Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.

Higuera, P., Lara, J., and Losada, I. (2013). Realistic wave generation and active wave absorption for Navier-Stokes models Application to OpenFOAM ®. *Coastal Engineering, Elsevier*, 71:102–118.

Hirt, C. and Nichols, B. (1981). Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201 – 225.

Hofland, B., Diamantidou, E., Steeg, P. V., and Meys, P. (2015). Wave runup and wave overtopping measurements using a laser scanner. 106:20–29.

Hofland, B., Ivo, W., and Paul, V. S. (2014). Short test durations for wave overtopping experiments.

Holthuijsen, L. (2007). *Waves in Oceanic and Coastal Waters*. Cambridge University Press.

Hsu, T., Sieh, H., Sai, T., and OU, C. (2015). Coupling vod/plic and embedding method for simulating wave breaking on a sloping beach. *Journal of Marine Science and Technology*, 23 (4):498–507.

Iribarren, R. and Nogales, C. (1949). Protection des ports. *XVII International Navigation Congress*. Lisbon, Section II-4.

Jacobsen, N. (2011). *A Full Hydro- and Morphodynamic Description of Breaker Bar Development*. PhD thesis.

Jacobsen, N. G., Fredsoe, J., and Jensen, J. H. (2014). Formation and development of a breaker bar under regular waves. part 1: Model description and hydrodynamics. *Coastal Engineering*, 88:182 – 193.

Jacobsen, N. G., Fuhrman, D. R., and Fredsøe, J. (2012). A Wave Generation Toolbox for the Open-Source CFD Library: OpenFoam®. *International Journal for Numerical Methods in Fluids*, 70(9):1073–1088.

Jacobsen, N. G., van Gent, M. R., Capel, A., and Borsboom, M. (2018). Numerical prediction of integrated wave loads on crest walls on top of rubble mound structures. *Coastal Engineering*, 142:110 – 124.

Keller, J. (1963). Tsunamis-waters waves produced by earthquakes. *International Union of Geodesy, Geophysics, Monogram*, 24:154–166.

Kharif, C. and Touboul, J. (2010). Under which conditions the Benjamin-Feir instability may spawn an extreme wave event: A fully nonlinear approach. *European Physical Journal: Special Topics*, 185(1):159–168.

Larsen, B. (2018). *Tsunami-Seabed Interactions*. PhD thesis.

Larsen, B. and Fuhrman, D. (2018). On the over-production of turbulence beneath surface waves in reynolds-averaged navier-stokes models. *Journal of Fluid Mechanics*, 853:419–460.

Larsen, B., Fuhrman, D., and Roenby, J. (2018). Performance of interfoam on the simulation of progressive waves. *arXiv:1804.01158 [physics.flu-dyn]*.

Li, L. and Dalrymple, R. (1998). Instabilities of the undertow. *Journal of Fluid Mechanics*, 369:175–190.

Loffredo, L., Schulz, D., and Wilkens, J. (2007). Design parameters for coastal dikes. *Environmental Fluid Mechanics*, 7(6):469–479.

Lubin, P., Vincent, S., Abadie, S., and Caltagirone, J. P. (2006). Three-dimensional Large Eddy Simulation of air entrainment under plunging breaking waves. *Coastal Engineering*, 53(8):631–655.

Madsen, P. A., Sorensen, O. R., and Schaffer, H. (1997). Surf zone dynamics simulated by a Boussinesq type model: Part I. Model description and cross-shore motion of regular waves. *Coastal Engineering, Elsevier*, 32:255–287.

Mansard, E. and Funke, E. (1980). The measurement of incident and reflected spectra using a least squares method. *Coastal Engineering*, pages 154–172.

Mayer, S. and Madsen, P. (2000). Simulation of breaking waves in the surf zone using a navier-stokes solver. In *International Conference in Coastal Engineering*.

Melville, W. K. (1982). The instability and breaking of deep water waves. *Journal of Fluid Mechanics*, 115:165–185.

Miche, R. (1944). Mouvements ondulatoires de la mer en profondeur constante ou decroissante. *Ecole nationale des ponts et chaussees*.

Miquel, A. M., Kamath, A., Alagan Chella, M., Archetti, R., and Bihs, H. (2018). Analysis of different methods for wave generation and absorption in a cfd-based numerical wave tank. *Journal of Marine Science and Engineering*, 6(2).

Munk, W. and Wimbush, M. (1969). A rule of thumb for wave breaking over sloping beaches. *Oceanology*, 6:56–59.

Ozdemir, C. E., Hsu, T.-J., and Balachandar, S. (2013). Direct numerical simulations of instability and boundary layer turbulence under a solitary wave. *Journal of Fluid Mechanics*, 731:545–578.

Pengzhi, L. and Philip, L. (1998). A numerical study of breaking waves in the surf zone. *Journal of Fluid Mechanics*, 359:239–264.

Peregrine, D. (1983). Breaking waves on beaches. *Annual Review of Fluid Mechanics*, 15:149–178.

Persson, P.-O. and Strang, G. (2003). Smoothing by savitzky-golay and legendre filters. In *Mathematical systems theory in biology, communications, computation, and finance*, pages 301–315. Springer.

Pocklington, M. (1921). Standing waves parallel to a plane beach. *Proceedings of Cambridge Philosophical Society*, 20:308–310.

Roelvink, D. J. and Stive, M. (1989). Bar-generating cross-shore flow mechanisms on a beach. *Journal of Geophysical Research*, 94:4785–4800.

Røenby, J., Bredmose, H., and Jasak, H. (2016). A computational method for sharp interface advection. *Royal Society Open Science*, 3:160405.

Rusche, H. (2002). *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. PhD thesis.

S Longuet-Higgins, M. and D. Cokelet, E. (1976). The deformation of steep surface waves on water. i. a numerical method of computation. *Proc. Roy. Soc. Lond. A*, 350:1–26.

Schiereck, G. (2016). *Introduction to Bed, Bank and Shore protection*. Delft Academic Press.

Scott, N. V., Hsu, T.-J., and Cox, D. (2009). Steep wave, turbulence, and sediment concentration statistics beneath a breaking wave field and their implications for sediment transport. *Continental Shelf Research, Elsevier*, 29:2303–2317.

Serio, F. D. and Mossa, M. (2006). Experimental study on the hydrodynamics of regular breaking waves. *Coastal Engineering*, 53(1):99 – 113.

Speziale, G., Abid, R., and Anderson, C. (1990). A critical evaluation of two-equation models for near wall turbulence. *AIAA Paper*, 90(1481).

Stansby, P. and Feng, T. (2004). Surf zone wave overtopping a trapezoidal structure: 1-D modelling and PIV comparison. *Coastal Engineering, Elsevier*, 51:483–500.

Stive, M. (1980). Velocity and pressure field of spilling breakers. *XVII Coastal Engineering Conference, Sydney, Australia*, pages 547–566.

Stokes, G. (1847). On the theory of oscillatory waves. *Transactions of the Cambridge Philosophical Society*, 8:441.

Svendsen, A. and Putrevu, U. (1995). Surf-zone hydrodynamics. *Center for Applied Coastal Research Ocean Engineering Laboratory University of Delaware*. Research Report Number: CACR-95-02.

Svendsen, I. (1984). Wave heights and set-up in a surf zone. *Coastal Engineering*, 8(4):303 – 329.

Svendsen, I. A. (1987). Analysis of Surf Zone Turbulence. 92(2):5115–5124.

Taylor, P. H. and Williams, B. A. (2019). Wave statistics for intermediate depth water — newwaves and symmetry. 126(February 2004).

Ting, F. and Kirby, J. (1994). Observation of undertow and turbulence in a laboratory surf zone. *Coastal Engineering*, 24, Issues 1–2,:51–80.

Tong, F. (2014). *Numerical simulations of steady and oscillatory flows around multiple cylinders*. PhD thesis.

Tromans, P. S., Anaturk, A. R., and Hagemeijer, P. (1991). A new model for the kinematics of large ocean waves application as a design wave. 8(August):11–16.

Van der Meer, J. (2002). Wave Run-up and Wave Overtopping at Dikes. *Technical Advisory Committee on Flood Defence*.

van Mierlo, F. (2014). Numerical modelling of wave penetration in ports. *Delft University of Technology*.

Vandebeek, I., Gruwez, V., Altomare, C., Suzuki, T., Vanneste, D., Roo, S. D., Toorman, E., and Troch, P. (2018). Towards an efficient and highly accurate coupled numerical modeling approach for wave interactions with a dike on a very shallow foreshore. *Proceedings of the 7th International Conference on the Application of Physical Modeling in Coastal and Port Engineering and Science (Coastalab18)*.

Watanabe, Y., Saeki, H., and Hosking, R. J. (2005). Three-dimensional vortex structures under breaking waves. *Journal of Fluid Mechanics*, 545:291–328.

Whittaker, C. N., Fitzgerald, C. J., Raby, A. C., Taylor, P. H., and Borthwick, A. G. L. (2018). Extreme coastal responses using focused wave groups : Overtopping and horizontal forces exerted on an inclined seawall. *Coastal Engineering*, 140(July):292–305.

Wilcox, D. (2006). Turbulence modeling for cfd. *3rd edition, DCW Industries, Inc., La Canada CA*.

Yuen, H. C. and Ferguson, W. E. (1978). Relationship between benjamin-feir instability and recurrence in the nonlinear schrödinger equation. *Physics of Fluids*, 21(8):1275–1278.

Zhou, Z., Sangermano, J., Hsu, T.-J., and Ting, F. C. (2014). A numerical investigation of wave-breaking-induced turbulent coherent structure under a solitary wave. *Journal of Geophysical Research: Oceans*, 119:6952–6973.

Zhou, Z., Tian-Jian, H., Daniel, C., and Xiaofeng, L. (2017). Large-eddy simulation of wave-breaking induced turbulent coherent structures and suspended sediment transport on a barred beach. *Journal of Geophysical Research: Oceans*, 122(1):207–235.

Zijlema, M., Stelling, G., and Smit, P. (2011). An operational public domain code for simulating wave fields and rapidly varied flows in coastal waters. *Coastal Engineering*, 58:992–1012.

Appendices

A SOURCE CODE FOR WAVEFLOW

OUTLINE

This chapter presents the integration of *waves2Foam* and *isoAdvection* libraries within OpenFOAM. The source code along with the compilation procedure will be discussed.

In order to sucessfully integrate the functionality of *waveFoam* and *isoAdvection*, the following steps can be followed. It is noted that the current version of *isoAdvection* does not support the use of *nAlphaSubCycles* > 1 and *nOuterCorrectors* > 1. This is due a bug in the code as pointed out by Johan Rønby[1].

The following steps can be followed to sucessfully compile *waveFoam* with *isoAdvection*. Once the correct *OpenFOAM* and *waves2Foam* environments are initialized, navigate to the location where the solvers are located. Generally, this should be at */username-v1806/applications/utilities/waves2Foam/applications/solvers/solvers1806_PLUS*. Make a directory structure as described below.

```
waveFlow/
 ├── alphaControls.H
 ├── alphaCourantNo.H
 ├── alphaEqn.H
 ├── alphaEqnSubCycle.H
 ├── alphaSuSp.H
 ├── correctPhi.H
 ├── createAlphaFluxes.H
 ├── createFields.H
 ├── initCorrectPhi.H
 ├── pEqn.H
 ├── rhofs.H
 ├── setDeltaT.H
 ├── setRDeltaT.H
 ├── UEqn.H
 ├── Make
 │   ├── files
 │   └── options
 └── waveFlow.C
```

Using the listings below, the solver files can be created. After this issuing the command *wmake* should be able to compile the new solver named *waveFlow*.

Listing A.1: alphaControls.H input file

```
const dictionary& alphaControls = mesh.solverDict(alpha1.name());

label  nAlphaSubCycles(readLabel(alphaControls.lookup("nAlphaSubCycles")));
```

Listing A.2: alphaCourantNo.H input file

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2017 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
```

[1]Personal Communication, Date: 27/April/2019

```
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Global
    alphaCourantNo

Description
    Calculates and outputs the mean and maximum Courant Numbers.

\*---------------------------------------------------------------------------*/

scalar maxAlphaCo
(
    readScalar(runTime.controlDict().lookup("maxAlphaCo"))
);

scalar alphaCoNum = 0.0;
scalar meanAlphaCoNum = 0.0;

if (mesh.nInternalFaces())
{
    scalarField sumPhi
    (
        mixture.nearInterface()().primitiveField()
       *fvc::surfaceSum(mag(phi))().primitiveField()
    );

    alphaCoNum = 0.5*gMax(sumPhi/mesh.V().field())*runTime.deltaTValue();

    meanAlphaCoNum =
        0.5*(gSum(sumPhi)/gSum(mesh.V().field()))*runTime.deltaTValue();
}

Info<< "Interface Courant Number mean: " << meanAlphaCoNum
    << " max: " << alphaCoNum << endl;

// ************************************************************************* //
```

Listing A.3: alphaEqn.H input file

```
// If there are more than one outer corrector, we use a mixture of old and
// new U and phi for propagating alpha1 in all but the first outer iteration
if (!pimple.firstIter())
{
    // We are recalculating alpha1 from the its old time value
    alpha1 = alpha1.oldTime();
    // Temporarily storing new U and phi values in prevIter storage
    U.storePrevIter();
    phi.storePrevIter();

    // Overwriting new U and phi values with mixture of old and new values
    phi = 0.5*(phi + phi.oldTime());
    U = 0.5*(U + U.oldTime());
}

// Update alpha1
advector.advect();

// Update rhoPhi
rhoPhi = advector.getRhoPhi(rho1, rho2);

alpha2 = 1.0 - alpha1;

if (!pimple.firstIter())
{
    // Restoring new U and phi values temporarily saved in prevIter() above
    U = U.prevIter();
    phi = phi.prevIter();
}

Info<< "Phase-1 volume fraction = "
    << alpha1.weightedAverage(mesh.Vsc()).value()
    << "  Min(" << alpha1.name() << ") = " << min(alpha1).value()
    << "  Max(" << alpha1.name() << ") - 1 = " << max(alpha1).value() - 1
    << endl;
```

Listing A.4: alphaEqnSubCycle.H input file

```
if (nAlphaSubCycles > 1)
{
    dimensionedScalar totalDeltaT = runTime.deltaT();
    surfaceScalarField rhoPhiSum
    (
        IOobject
        (
            "rhoPhiSum",
            runTime.timeName(),
            mesh
        ),
        mesh,
        dimensionedScalar(rhoPhi.dimensions(), Zero)
    );

    tmp<volScalarField> trSubDeltaT;

    for
    (
        subCycle<volScalarField> alphaSubCycle(alpha1, nAlphaSubCycles);
        !(++alphaSubCycle).end();
    )
    {
        #include "alphaEqn.H"
        rhoPhiSum += (runTime.deltaT()/totalDeltaT)*rhoPhi;
    }

    rhoPhi = rhoPhiSum;
}
else
{
    #include "alphaEqn.H"
}

rho == alpha1*rho1 + alpha2*rho2;
```

Listing A.5: alphaSuSp.H input file

```
zeroField Su;
zeroField Sp;
zeroField divU;
```

Listing A.6: correctPhi.H input file

```
CorrectPhi
(
    U,
    phi,
    p_rgh,
    dimensionedScalar("rAUf", dimTime/rho.dimensions(), 1),
    geometricZeroField(),
    pimple
);

#include "continuityErrs.H"
```

Listing A.7: createAlphaFluxes.H input file

```
IOobject alphaPhi10Header
(
    "alphaPhi10",
    runTime.timeName(),
    mesh,
    IOobject::READ_IF_PRESENT,
    IOobject::AUTO_WRITE
);

const bool alphaRestart =
    alphaPhi10Header.typeHeaderOk<surfaceScalarField>(true);

// MULES flux from previous time-step
surfaceScalarField alphaPhi10
(
    alphaPhi10Header,
    phi*fvc::interpolate(alpha1)
);

// MULES Correction
tmp<surfaceScalarField> talphaPhi1Corr0;
```

Listing A.8: createFields.H input file

```
#include "readGravitationalAcceleration.H"
#include "readWaveProperties.H"
#include "createExternalWaveForcing.H"

//#include "createRDeltaT.H"

Info<< "Reading field p_rgh\n" << endl;
volScalarField p_rgh
(
    IOobject
    (
        "p_rgh",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

```

```
#include "createPhi.H"


Info<< "Reading transportProperties\n" << endl;
immiscibleIncompressibleTwoPhaseMixture mixture(U, phi);

volScalarField& alpha1(mixture.alpha1());
volScalarField& alpha2(mixture.alpha2());

const dimensionedScalar& rho1 = mixture.rho1();
const dimensionedScalar& rho2 = mixture.rho2();


// Need to store rho for ddt(rho, U)
volScalarField rho
(
    IOobject
    (
        "rho",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT
    ),
    alpha1*rho1 + alpha2*rho2
);
rho.oldTime();


// Mass flux
surfaceScalarField rhoPhi
(
    IOobject
    (
        "rhoPhi",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    fvc::interpolate(rho)*phi
);


// Construct incompressible turbulence model
autoPtr<incompressible::turbulenceModel> turbulence
(
    incompressible::turbulenceModel::New(U, phi, mixture)
);

#include "readhRef.H"
#include "gh.H"


// volScalarField gh("gh", g & (mesh.C() - referencePoint ));
// surfaceScalarField ghf("ghf", g & (mesh.Cf() - referencePoint ));



volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    p_rgh + rho*gh
);

label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell
(
    p,
    p_rgh,
    pimple.dict(),
    pRefCell,
    pRefValue
);

if (p_rgh.needReference())
{
    p += dimensionedScalar
    (
        "p",
        p.dimensions(),
        pRefValue - getRefCellValue(p, pRefCell)
    );
    p_rgh = p - rho*gh;
}

mesh.setFluxRequired(p_rgh.name());
mesh.setFluxRequired(alpha1.name());

// MULES compressed flux is registered in case scalarTransport FO needs it.
/* surfaceScalarField alphaPhiUn
(
    IOobject
    (
        "alphaPhiUn",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("zero", phi.dimensions(), 0.0)
);
*/
```

```
#include "createMRF.H"


relaxationZone relaxing(mesh, U, alpha1);

isoAdvection advector(alpha1, phi, U);
```

Listing A.9: initCorrectPhi.H input file

```
tmp<volScalarField> rAU;

if (CorrectPhi)
{
    rAU = new volScalarField
    (
        IOobject
        (
            "rAU",
            runTime.timeName(),
            mesh,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        mesh,
        dimensionedScalar("rAU", dimTime/dimDensity, 1)
    );

  #include "correctPhi.H"
}
else
{
    CorrectPhi
    (
        U,
        phi,
        p_rgh,
        dimensionedScalar("rAUf", dimTime/rho.dimensions(), 1),
        geometricZeroField(),
        pimple
    );

    #include "continuityErrs.H"
}
```

Listing A.10: pEqn.H input file

```
{
    volScalarField rAU("rAU", 1.0/UEqn.A());
    surfaceScalarField rAUf("rAUf", fvc::interpolate(rAU));

    volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p_rgh));

    surfaceScalarField phiHbyA
    (
        "phiHbyA",
        fvc::flux(HbyA)
      + fvc::interpolate(rho*rAU)*fvc::ddtCorr(U, phi)
    );
    MRF.makeRelative(phiHbyA);
    adjustPhi(phiHbyA, U, p_rgh);

    surfaceScalarField phig
    (
        (
            mixture.surfaceTensionForce()
          - ghf*fvc::snGrad(rho)
        )*rAUf*mesh.magSf()
    );

    phiHbyA += phig;

    // Update the pressure BCs to ensure flux consistency
    constrainPressure(p_rgh, U, phiHbyA, rAUf, MRF);

    while (pimple.correctNonOrthogonal())
    {
        fvScalarMatrix p_rghEqn
        (
            fvm::laplacian(rAUf, p_rgh) == fvc::div(phiHbyA)
        );

        p_rghEqn.setReference(pRefCell, getRefCellValue(p_rgh, pRefCell));

        p_rghEqn.solve(mesh.solver(p_rgh.select(pimple.finalInnerIter ())));

        if (pimple.finalNonOrthogonalIter())
        {
            phi = phiHbyA - p_rghEqn.flux();

            p_rgh.relax ();

            U = HbyA + rAU*fvc::reconstruct((phig - p_rghEqn.flux())/rAUf);
            U.correctBoundaryConditions();
            fvOptions.correct(U);
        }
    }

    #include "continuityErrs.H"

    p == p_rgh + rho*gh;

    if (p_rgh.needReference())
    {
        p += dimensionedScalar
        (
            "p",
            p.dimensions(),
            pRefValue − getRefCellValue(p, pRefCell)
        );
        p_rgh = p − rho∗gh;
    }
}
```

Listing A.11: rhofs.H input file

```
const dimensionedScalar& rho1f(rho1);
const dimensionedScalar& rho2f(rho2);
```

Listing A.12: setDeltaT.H input file

```
/*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−∗\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011−2017 OpenFOAM Foundation
     \\/     M anipulation  |
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Global
    setDeltaT

Description
    Reset the timestep to maintain a constant maximum courant Number.
    Reduction of time−step is immediate, but increase is damped to avoid
    unstable oscillations.

\∗−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−∗/

if (adjustTimeStep)
{
    scalar maxDeltaTFact =
        min(maxCo/(CoNum + SMALL), maxAlphaCo/(alphaCoNum + SMALL));

    scalar deltaTFact = min(min(maxDeltaTFact, 1.0 + 0.1∗maxDeltaTFact), 1.2);

    runTime.setDeltaT
    (
        min
        (
            deltaTFact∗runTime.deltaTValue(),
            maxDeltaT
        )
    );

    Info<< "deltaT = " << runTime.deltaTValue() << endl;
}

// ∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗∗ //
```

Listing A.13: UEqn.H input file

```
    MRF.correctBoundaryVelocity(U);

    fvVectorMatrix UEqn
    (
        fvm::ddt(rho, U) + fvm::div(rhoPhi, U)
      + MRF.DDt(rho, U)
      + turbulence−>divDevRhoReff(rho, U)
     ==
        fvOptions(rho, U)
    );

    UEqn.relax();

    fvOptions.constrain(UEqn);

    if (pimple.momentumPredictor())
    {
        solve
        (
            UEqn
         ==
            fvc::reconstruct
            (
                (
                    mixture.surfaceTensionForce()
                  − ghf∗fvc::snGrad(rho)
                  − fvc::snGrad(p_rgh)
                ) ∗ mesh.magSf()
            )
        );

        fvOptions.correct(U);
    }
```

Listing A.14: waveFlow.C input file

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2017 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Application
    interFoam

Group
    grpMultiphaseSolvers

Description
    Solver for 2 incompressible, isothermal immiscible fluids using a VOF
    (volume of fluid) phase-fraction based interface capturing approach.

    The momentum and other fluid properties are of the "mixture" and a single
    momentum equation is solved.

    Turbulence modelling is generic, i.e. laminar, RAS or LES may be selected.

    For a two-fluid approach see twoPhaseEulerFoam.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
//#include "CMULES.H"

// Include the isoAdvection header file
#include "isoAdvection.H"

//#include "EulerDdtScheme.H"
//#include "localEulerDdtScheme.H"
//#include "CrankNicolsonDdtScheme.H"
#include "subCycle.H"
#include "immiscibleIncompressibleTwoPhaseMixture.H"
#include "turbulentTransportModel.H"
#include "pimpleControl.H"
#include "fvOptions.H"
#include "CorrectPhi.H"
#include "fvcSmooth.H"

#include "relaxationZone.H"
#include "externalWaveForcing.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{

    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"

    #include "createControl.H"
    #include "createTimeControls.H"
    #include "initContinuityErrs.H"

    #include "createFields.H"
    //#include "createAlphaFluxes.H"
    #include "createFvOptions.H"
    #include "correctPhi.H"
    #include "postProcess.H"

    turbulence->validate();

    #include "readTimeControls.H"
    #include "CourantNo.H"
    #include "setInitialDeltaT.H"


    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nStarting time loop\n" << endl;

    while (runTime.run())
    {
        #include "readTimeControls.H"

        #include "CourantNo.H"
        #include "alphaCourantNo.H"
        #include "setDeltaT.H"

        runTime++;

        Info<< "Time = " << runTime.timeName() << nl << endl;

        externalWave->step();

        // --- Pressure-velocity PIMPLE corrector loop
        while (pimple.loop())
        {
            #include "alphaControls.H"
            #include "alphaEqnSubCycle.H"

            relaxing.correct ();

            mixture.correct ();

            if  (pimple.frozenFlow())
            {
                continue;
            }

            #include "UEqn.H"

            // --- Pressure corrector loop
            while (pimple.correct())
            {
                #include "pEqn.H"
            }

            if  (pimple.turbCorr())
            {
                turbulence->correct();
            }
        }

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "   ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    // Close down the external wave forcing in a nice manner
    externalWave->close();

    Info<< "End\n" << endl;

    return 0;
}


// ************************************************************************* //
```

Listing A.15: files input information

```
waveFlow.C

EXE = $(WAVES_USER_APPBIN)/waveFlow
```

Listing A.16: options input file

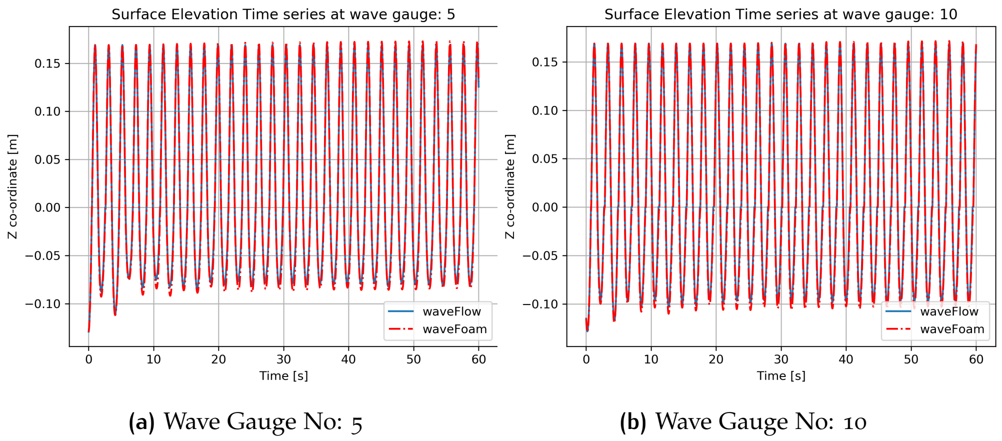
```
EXE_INC = \
    -I$(LIB_SRC)/transportModels/twoPhaseMixture/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels/interfaceProperties/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels/immiscibleIncompressibleTwoPhaseMixture/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -DOFVERSION=1712 \
    -DEXTBRANCH=0 \
    -DOFPLUSBRANCH=1 \
    -DXVERSION=$(WAVES_XVERSION) \
    -I$(WAVES_SRC)/waves2Foam/lnInclude \
    -I$(WAVES_SRC)/waves2FoamSampling/lnInclude \
    -I$(WAVES_SRC)/waves2FoamOvertopping/lnInclude \
    -I$(WAVES_GSL_INCLUDE)

EXE_LIBS = \
    -limmiscibleIncompressibleTwoPhaseMixture \
    -lturbulenceModels \
    -lincompressibleTurbulenceModels \
    -lfiniteVolume \
    -lfvOptions \
    -lmeshTools \
    -lsampling \
    -L$(WAVES_LIBBIN) \
    -lwaves2Foam \
    -lwaves2FoamSampling \
    -lwaves2FoamOvertopping \
    -L$(WAVES_GSL_LIB) \
    -lgsl \
    -lgslcblas
```

B CALIBRATION TEST RESULTS

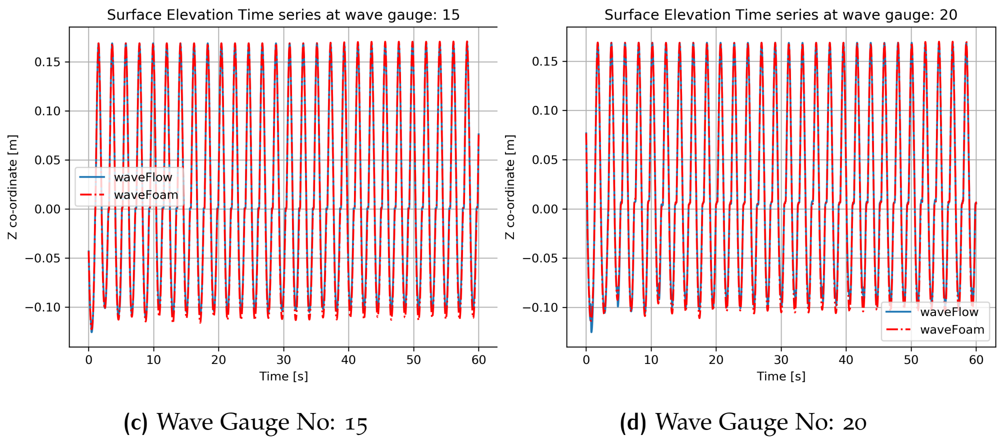
OUTLINE

This chapter presents the results of the calibration test which is based upon a simple sloping beach with a vertical wall in the end. This conceptual model does not reply on any experimental investigation, however, some qualitative results have been presented in this section. The additional results have been included in this part of the report for sake of completeness.



(a) Wave Gauge No: 5

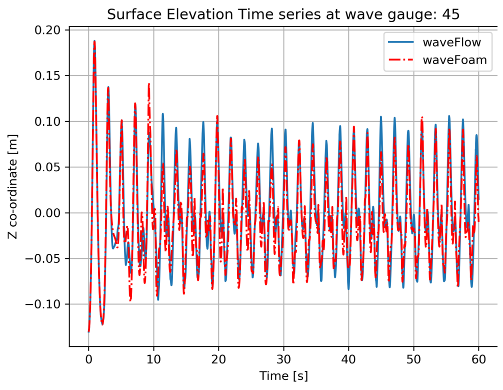
(b) Wave Gauge No: 10



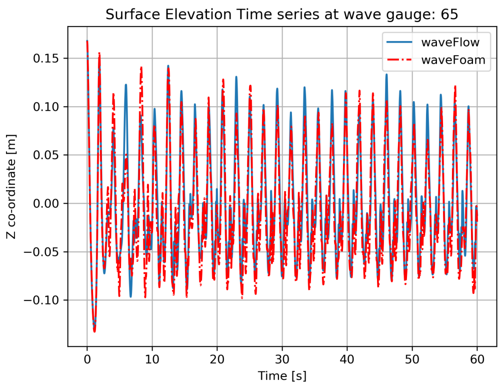
(c) Wave Gauge No: 15

(d) Wave Gauge No: 20

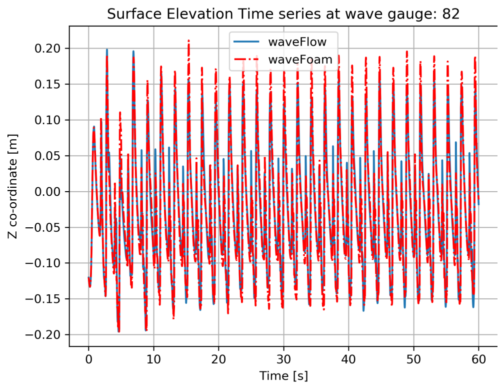
Figure B.1: Surface elevation time series comparison within the relaxation zone. Approximately 20 wave gauge's are present within the relaxation zone (See Figure 3.3)



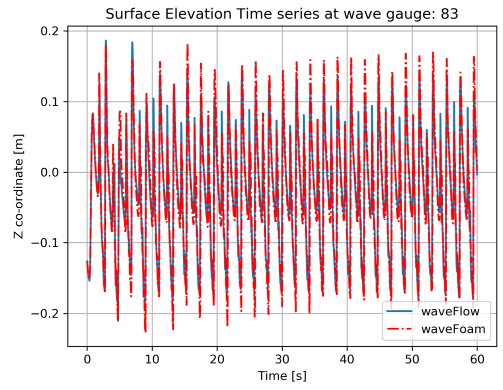
(a) Wave Gauge No: 45



(b) Wave Gauge No: 65



(c) Wave Gauge No: 82



(d) Wave Gauge No: 83

Figure B.2: Surface elevation time series comparison outside the relaxation zone. Wave gauge 82 and 83 are close to the vertical structure as seen in Figure 3.2

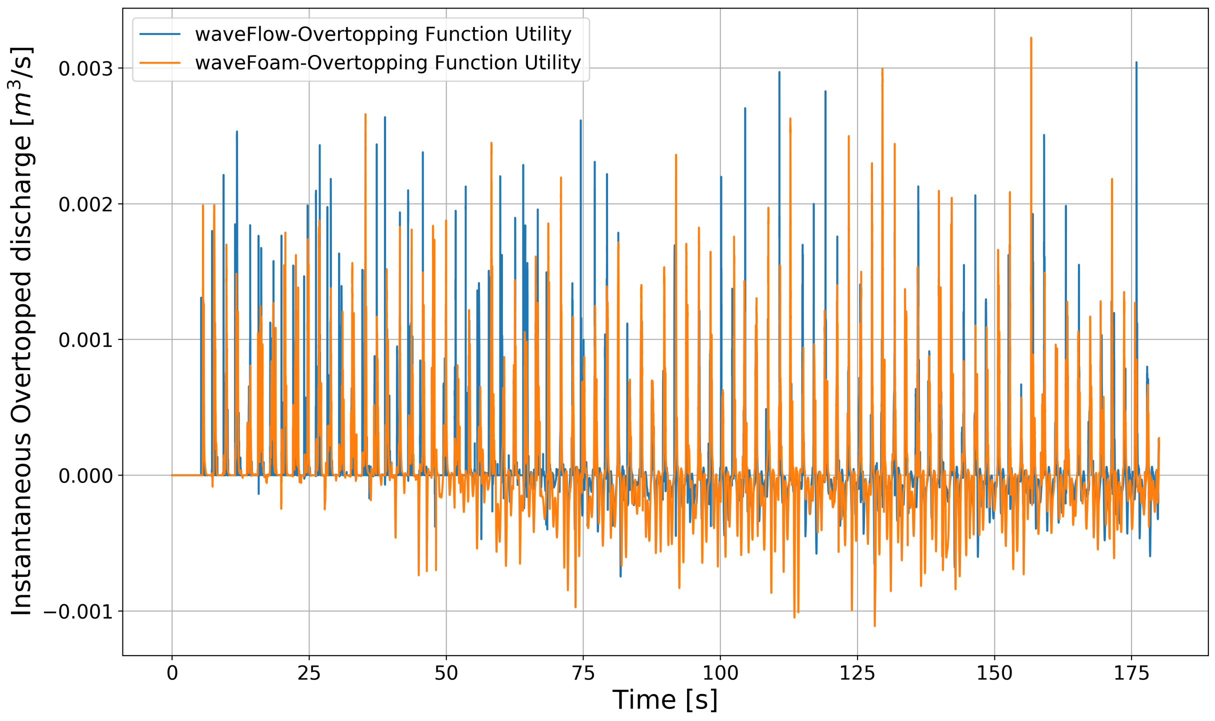


Figure B.3: Overtopping time series for *waveFlow* and *waveFoam*.

C TING AND KIRBY SETUP

OUTLINE

This chapter details the numerical model setup used for the Ting and Kirby [1994] numerical simulations. The numerical model setup are described in the initial sections of this chapter, followed by some additional results and discussions.

The case is organised as seen in the directory structure below. The conventional `OpenFOAM` case structure has been utilized in the setup. All the files concerning the model setup can be found in the listings below.

```
TingAndKirbyCase/
├── 0/
│   ├── alpha.water
│   ├── nut
│   ├── omega
│   ├── p_rgh
│   └── U
├── constant/
│   ├── environmentalProperties
│   ├── postProcessingProperties
│   ├── probeDefinitions
│   ├── transportProperties
│   ├── turbulenceProperties
│   └── waveProperties.input
└── system/
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    └── probeData
```

Listing C.1: alpha.water input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            waveAlpha;
        refValue        uniform 0
        refGradient     uniform 0;
        valueFraction   uniform 1;
        value           uniform 0;
    }
    outlet
    {
        type            zeroGradient;
    }
    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    bottom
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}


// ************************************************************************* //
```

Listing C.2: nut input file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
     inlet
    {
        type            zeroGradient;
    }

    outlet
    {
    type            nutkRoughWallFunction;
    Ks        uniform 0;  //Sand grain roughness height (0 for smooth walls)
    Cs        uniform 0.5;     //Roughness constant (~0.5 to 1.0)
    value           $internalField;
    }

    bottom
    {
        type           nutkRoughWallFunction;
    //Nikurdse roughness for plywood as specified in Larsen 2018
    Ks        uniform 1e-04; //Sand grain roughness height (0 for smooth walls)
    Cs        uniform 0.5;     //Roughness constant (~0.5 to 1.0)
    value           $internalField;

    }

    atmosphere
    {
        type            calculated;
        value           uniform 0;
    }

    frontBack
    {
        type            empty;
    }
}


// ************************************************************************* //
```

Listing C.3: omega input file

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform 1.977;
```

```

boundaryField
{
    bottom
    {
        type            omegaWallFunction;
        //Only an initial guess to speed up the solution
        //Can use different approaches to save computational time
    value       $internalField;
    }

    atmosphere
    {
         type           inletOutlet;
         inletValue     $internalField;
         value      $internalField;
    }

    frontBack
    {
        type            empty;
    }

     inlet
    {
        type            zeroGradient;
    }

    outlet
    {
    $bottom
    }
}


// ************************************************************************* //
```

Listing C.4: p_rgh input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
     inlet
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 0;
    }
    outlet
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 0;
    }
    atmosphere
    {
        type            totalPressure;
        rho             rho;
        psi             none;
        gamma           1;
        p0              uniform 0;
        value           uniform 0;
    }
    bottom
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 0;
    }
    frontAndBack
    {
        type            empty;
    }
}


// ************************************************************************* //
```

Listing C.5: U input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];


internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            waveVelocity;
        refValue        uniform (0 0 0);
        refGradient     uniform (0 0 0);
        valueFraction   uniform 1;
        value           uniform (0 0 0);
    }

    outlet
    {
        type            noSlip;
    }

    atmosphere
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }

    bottom
    {
        type            noSlip;
    }

    frontAndBack
    {
        type            empty;
    }
}


// ************************************************************************* //
```

Listing C.6: environmentalProperties input file

```
/*---------------------------------*- C++ -*-----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      environmentalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

g               g [0 1 -2 0 0 0 0] (0 0 -9.81);

// ************************************************************************* //
```

Listing C.7: postProcessingProperties input file

```
/*---------------------------------*- C++ -*-----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      postProcessingProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

deleteParentOutputDirectory false;

regularSpectrum
{
    callName surfaceElevation;

    removeDuplicate true;
    inputDir surfaceElevationAnyName;

    deltaT 0.0005;

    tMin    0;
    tMax    253;

    actionList ( interpolateSurfaceElevation );

    nFreq 10;
    period 2.0;
    allDataSets true;
}

// ************************************************************************* //
```

Listing C.8: transportProperties input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water air);

water
{
    transportModel Newtonian;
    nu              1e-06;
    rho             rho [1 -3 0 0 0] 1000;
}

air
{
    transportModel Newtonian;
    nu              1.48e-05;
    rho             rho [1 -3 0 0 0] 1;
}

sigma           0.07;   //Can be set to 0

// ************************************************************************* //
```

Listing C.9: turbulenceProperties input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RAS;

RAS
{
    RASModel kOmegaStab;

    turbulence on;

    printCoeffs on;

    kOmegaStabCoeffs
    {
        // It was shown that lambda1 = 0; does not results in a spilling wave
        lambda1     0.875;  //Based on Larsen 2018

        // Stabilisation factor
        lambda2     0.05;   //Based on Larsen 2018
    }
}


// ************************************************************************* //
```

Listing C.10: waveProperties.input input file

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class   dictionary;
    object  waveProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

seaLevel            0;

relaxationNames    ( inlet );

initializationName  inlet ;

inletCoeffs
{
    //Type of wave
    waveType   streamFunction;

    //Height of the wave at the boundary
    height       0.125;   // [m]

    //Depth at which he imposed wave enters the computational domain
    depth        0.4;     // [m]

    //Number of components
    N            30;  // [−]

    //Number of Iterations
    Niter        40;  // [−]

    //Phase of the wave
    phi          0;   // [radians]

    // Direction of the wave
    direction       (1 0 0);     // (x y z) propagation normal vector

    //Decision for information  specification
    specifyPeriod   true;

    //Period of the incoming wave
    period       2.0;     // [s]

    // Specify Euler boolean
    specifyEuler    false;

        //Stokes velocity at the inlet
        stokesVelocity   0;   // [m/s]

        // Specification  of Tsoft
        Tsoft    2;          //Equal to the wave period


    relaxationZone
    {
        relaxationScheme   Spatial;
        relaxationShape    Rectangular;
        beachType          Empty;
        relaxType          INLET;
        startX             ( −8 0 0 );
        endX               ( −4 0.1 0  );
        orientation        ( 1 0 0 );
    }
}




// ************************************************************************* //
```

Listing C.11: blockMeshDict input file

```
/*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*− C++ −*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1806                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*/
FoamFile
{
    version         2.0;
    format          ascii;
    class           dictionary;
    object          blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//This line will allow scaling of the dimensions. For example, 0.01 will convert all the units to cm scale
scale 1;

//Resolution in block 1

resX1 800;
resZ1 80;

//Resolution in block 1
resX2 1550;
resZ2 80;

vertices
(
    //Block 1
    (−8 0 −0.4)          //Point 0
    (0 0 −0.4)           //Point 1
    (0 0 0.4)                //Point 2
    (−8 0 0.4)           //Point 3

    (−8 0.1 −0.4)                //Point 4
        (0 0.1 −0.4)                 //Point 5
        (0 0.1 0.4)                  //Point 6
        (−8 0.1 0.4)                 //Point 7

    //Block 2
    (15.95 0 0.0557)          //Point 8
    (15.95 0 0.8557)          //Point 9

    (15.95 0.1 0.0557)                //Point 10
        (15.95 0.1 0.8557)                //Point 11

);

blocks
(
    hex (0 1 5 4 3 2 6 7) ($resX1 1 $resZ1) simpleGrading (1 1 14)

    hex (1 8 10 5 2 9 11 6) ($resX2 1 $resZ2) simpleGrading (2 1 14)
);

edges
(


);

boundary
(
     inlet
    {
        type    patch;
        faces
        (
            (0 4 7 3)
        );
    }

    outlet
    {
        type    wall;
        faces
        (
            (8 10 11 9)
        );
    }

    atmosphere
    {
        type    patch;
        faces
        (
            (2 9 11 6)
            (3 2 6 7)
        );
    }

    bottom
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (1 8 10 5)
        );
    }

    frontAndBack
    {
        type    empty;
        faces
        (
            (0 3 2 1)
            (4 7 6 5)
            (1 2 9 8)
            (5 6 11 10)
        );
    }

);

mergePatchPairs
(
);

// ************************************************************************* //
```

Listing C.12: controlDict input file

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

```

```
endTime          330;

deltaT           0.0001;

writeControl     adjustableRunTime;

writeInterval    0.5;

purgeWrite       0;

writeFormat      ascii ;

writePrecision   6;

timeFormat       general;

timePrecision    6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo            0.05;

maxAlphaCo       0.05;

maxDeltaT        1;


//Include modified Turbulence Model Library

libs
(
    "libstabRASModels.so"
);


//Include functions for runtime postProcessing

functions
{
    //Function Definition Files

    //This file is generated using the command waveGaugeNProbes
    //Surface Elevation Include File
    #includeIfPresent "../waveGaugesNProbes/surfaceElevationAnyName_controlDict"

    //Probe data include file
    #include "probeData"

}

// ************************************************************************* //
```

Listing C.13: fvSchemes input file

```
/*---------------------------------*- C++ -*-----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default      Euler;
}

gradSchemes
{
    default          Gauss linear;
    grad(U)          Gauss linear;
    grad(alpha1)     Gauss linear;
}

divSchemes
{
    default none;

        div(rhoPhi,U)  Gauss linearUpwind Gauss linear phi;
        div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
        div(phi,alpha)  Gauss vanLeer;
        div(phirb,alpha) Gauss interfaceCompression;

    //Turbulence Terms
    div(rhoPhi,k)        Gauss Minmod;
        div(rhoPhi,omega) Gauss Minmod;

    div(phi,k)        Gauss Minmod;
        div(phi,omega) Gauss Minmod;
    div((muEff*dev(T(grad(U))))) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;

}

laplacianSchemes
{
    default          Gauss linear corrected;
}

```

```
interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha.water;
}

// ************************************************************************* //
```

Listing C.14: fvSolution input file

```
/*---------------------------------*- C++ -*-----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        isoFaceTol          1e-10;
        surfCellTol         1e-6;
        nAlphaBounds    3;
        snapTol             1e-12;
        clip                true;

        nAlphaSubCycles   1;
        //cAlpha is not used by isoAdvector but must be specified because interfacePropertes object reads it during construction.
    cAlpha          1;
    }

    pcorr
    {
    solver          PCG;
        tolerance           1e-10;
        relTol                  0.0;

        preconditioner      DIC;
    }

    pcorrFinal
    {
    solver          PCG;
        tolerance           1e-10;
        relTol          0.0;

        preconditioner      DIC;
    }

    p_rgh
    {
        solver          PCG;
        tolerance           1e-7;
        relTol              0;

        preconditioner      DIC;
    }

    p_rghFinal
    {
        solver          PCG;
        tolerance           1e-7;
        relTol              0;

        preconditioner   DIC;
    }

    U
    {
    solver          PBiCG;
        preconditioner      DILU;
        tolerance           1e-07;
        relTol              0;
    }

    UFinal
    {
    solver          PBiCG;
        preconditioner      DILU;
        tolerance           1e-09;
        relTol              0;
    }

    k
    {
    solver                  PBiCG;
        preconditioner  DILU;
```

```
            tolerance           1e−07;
            relTol              0;
        }

        kFinal
        {
        solver                  PBiCG;
            preconditioner      DILU;
            tolerance           1e−09;
            relTol             0;
        }

        omega
        {
        solver                  PBiCG;
            preconditioner      DILU;
            tolerance           1e−08;
            relTol              0;
        }

        omegaFinal
        {
        solver           PBiCG;
        preconditioner       DILU;
        tolerance            1e−08;
        relTol           0;
        }
}


PIMPLE
{
    momentumPredictor yes;
    nOuterCorrectors 1;
    nCorrectors     4;
    nNonOrthogonalCorrectors 1;
}

relaxationFactors
{
     fields
     {
     }
    equations
    {
        ".*"  1;
    }
}


// ************************************************************************* //
```

Listing C.15: probeData input file

```
/*---------------------------------*- C++ -*-----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.5                                   |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/



probeData
{
    //Define the number of points
    number 100;

    //Sampling library import
    functionObjectLibs ( "libsampling.so" );

    //Type of probe (set -> 1D sampling)
    type                sets;

        //Switch to enable
        enabled             true;

    // writeControl flag
        writeControl        adjustableRunTime;

        //Write interval period
        writeInterval       0.01;

    //Format of the data
    setFormat   raw;

    // Interpolation scheme
    interpolationScheme cellPoint;

    // Fields to be sampled along the lines
     fields
    (
        U
        p
        k
        omega
        alpha.water
    );

    sets
        (
            WG1 //Absolute location −1.265 m
            {
                type uniform;
                axis z;
                 start (−1.265 0.05 −0.4);
                end   (−1.265 0.05 0.125);
                nPoints $number;
            }

            WG2 //Absolute location 5.945 m
            {
                type uniform;
                axis z;
                start (5.945 0.05 −0.23);
                end   (5.945 0.05 0.2);
                nPoints $number;
            }

            WG3 //Absolute location 6.665 m
            {
                type uniform;
                axis z;
                start (6.665 0.05 −0.2095);
                end   (6.665 0.05 0.2);
                nPoints $number;
            }

            WG4 //Absolute location 7.275 m
            {
                type uniform;
                axis z;
                start (7.275 0.05 −0.1921);
                end   (7.275 0.05 0.2);
                nPoints $number;
            }

            WG5 //Absolute location 7.885  m
            {
                type uniform;
                axis z;
                start (7.885  0.05 −0.1747);
                end   (7.885  0.05 0.2);
                nPoints $number;
            }

            WG6 //Absolute location 8.495 m
            {
                type uniform;
                axis z;
                start (8.495 0.05 −0.1573);
                end   (8.495 0.05 0.2);
                nPoints $number;
            }

            WG7 //Absolute location 9.11 m
            {
                type uniform;
                axis z;
                start (9.11 0.05 −0.1397);
                end   (9.11 0.05 0.2);
                nPoints $number;
            }

            WG8 //Absolute location 9.725 m
            {
                type uniform;
                axis z;
                start (9.725 0.05 −0.1221);
                end   (9.725 0.05 0.2);
                nPoints $number;
            }

        );

}



// ************************************************************************* //
```

D PYTHON CODE USED FOR ANALYSIS

OUTLINE

This chapter includes the code listing for all the python codes used in the data analysis routines.

Listing D.1: Python script used for min-max-mean analysis for the validation studies for Ting and Kirby [1994]

```python
#Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#Signal processing library
from scipy.signal import argrelextrema

#Force python to use qt version of plotting
%matplotlib qt

#Loading the data required for the analysis
data = pd.read_csv('waveFlow/0/surfaceElevation.dat',sep='\t')

#Set the analysis bin window
n = 100 #Roughly equal to wavePeriod*samplingRate*0.5

#Set the location parameters by slicing the index before dropping of the data is done
location = data[:3]

# Initialize storage arrays for each of the variables
mean_eta = np.zeros(186)
maximum = np.zeros(186)
sdMax = np.zeros(186)
minimum = np.zeros(186)
sdMin = np.zeros(186)

#min-max analysis and look for the start and end of the database
for i in range(0,186):
    #Carrying out the min-max analysis
    data['min'] = data.iloc[argrelextrema(data['gauge_'+str(i)].values, np.less_equal, order=n)[0]]['gauge_'+str(i)]
    data['max'] = data.iloc[argrelextrema(data['gauge_'+str(i)].values, np.greater_equal, order=n)[0]]['gauge_'+str(i)]

    #Set start and end of the dataset
    start = np.where(np.isnan(data['min']) == False)[0][0]
    end = np.where(np.isnan(data['min']) == False)[0][-1]

    #Drop the first and the last datapoints to avoid bias for incorrect start/end of database
    myData = data[start:end]

    #Carry out the calculations
    mean_eta[i] = np.mean(myData['gauge_'+str(i)])

    #Compute the statistics
    maximum[i] = np.mean(myData['max'])
    sdMax[i] = np.std(myData['max'])
    minimum[i] = np.mean(myData['min'])
    sdMin[i] = np.std(myData['min'])

    #Print the current wave gauge
    print('Wave Gauge ',i,' done',end='\r')

#Saving the text file
np.savetxt('pythonCodeData/waveFlow/maximum.dat',maximum,delimiter='\t')
np.savetxt('pythonCodeData/waveFlow/minimum.dat',minimum,delimiter='\t')
np.savetxt('pythonCodeData/waveFlow/minSD.dat',sdMin,delimiter='\t')
np.savetxt('pythonCodeData/waveFlow/maxSD.dat',sdMax,delimiter='\t')
np.savetxt('pythonCodeData/waveFlow/meanSurfaceElevation_waveFlow.txt',mean_eta,delimiter='\t')
```

Listing D.2: Python script used to prepare the data for extraction and compiling of α_w

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 18 14:24:42 2019

@author: akshay
"""

#Importing the required libraries
import numpy as np
import pandas as pd

#Read the ls.dat file which contains an ordered list of directory values
timeListing = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/ls.dat',header=None,sep='\t',dtype=str)

```

```
for gaugeLoop in range(1,9):

    #User input for which wave gauge is to be analyzed
    waveGaugeNumber = gaugeLoop

    #Read the data to store the y location of the selected wave gauge
    location = np.loadtxt('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
    /waveFlow/undertow/60.1/WG'+str(waveGaugeNumber)+'_alpha.water_k_omega_p.xy',dtype=str,usecols=[0])

    #Create a new empty dataframe which has y values as columns and add a time column
    sampledData = pd.DataFrame(columns=[location])

    #Adding the time column in the first location
    sampledData.insert(0,'Time',None)

    #Looping through all the time directories
    for i in timeListing.iloc [:,0]:

        #Set the time value for the sampledData dataframe
        sampledData.loc[i,'Time'] = i

        #Create a dataLocation variable which changes as the time directory value changes
        dataLocation = '/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/undertow/'+str(i)+'/'

        #Read the data location file for a specified column only
        myData = pd.read_csv(str(dataLocation)+'WG'+str(waveGaugeNumber)+'_alpha.water_k_omega_p.xy',header=None,sep='\t',usecols=[0,1])

        if (waveGaugeNumber == 6):
            lengthOfArray = 99
        else:
            lengthOfArray = 100

        #Looping through the 100 data values and storing the values for the given parameter
        for j in range(lengthOfArray):
            #Computing the velocity and saving it
            sampledData.loc[i,location[j]] = myData.iloc[j,1]
        #Print the current i index
        print("Time ",i," done! for waveGauge:",waveGaugeNumber)

    #Saving the data
    sampledData.to_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
    /pythonCodeData/waveFlow/undertow/undertow_alpha_waveGauge_'+str(waveGaugeNumber)+'.dat',sep='\t')
```

Listing D.3: Python script used to prepare the data for extraction and compiling of k

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 18 14:24:42 2019

@author: akshay
"""

#Importing the required libraries
import numpy as np
import pandas as pd

#Read the ls.dat file which contains an ordered list of directory values
timeListing = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/
TINGandKIRBY/waveFlow/ls.dat',header=None,sep='\t',dtype=str)

#User input for which wave gauge is to be analyzed
waveGaugeNumber = 8

#Read the data to store the y location of the selected wave gauge
location = np.loadtxt('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/
TINGandKIRBY/waveFlow/undertow/60.1/WG'+str(waveGaugeNumber)+'_alpha.water_k_omega_p.xy',dtype=str,usecols=[0])

#Create a new empty dataframe which has y values as columns and add a time column
sampledData = pd.DataFrame(columns=[location])

#Adding the time column in the first location
sampledData.insert(0,'Time',None)

#Looping through all the time directories
for i in timeListing.iloc [:,0]:

    #Set the time value for the sampledData dataframe
    sampledData.loc[i,'Time'] = i

    #Create a dataLocation variable which changes as the time directory value changes
    dataLocation = '/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/undertow/'+str(i)+'/'

    #Read the data location file for a specified column only
    myData = pd.read_csv(str(dataLocation)+'WG'+str(waveGaugeNumber)+'_alpha.water_k_omega_p.xy',header=None,sep='\t',usecols=[1,2])

    lengthOfArray = len(myData)

    #Looping through the 100 data values and storing the values for the given parameter
    for j in range(lengthOfArray):
        #Computing the velocity and saving it
        sampledData.loc[i,location[j]] = myData.iloc[j,1]
    #Print the current i index
    print("Time ",i," done!")

#Saving the data
sampledData.to_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/undertow_k_waveGauge_'+str(waveGaugeNumber)+'.dat',sep='\t')
```

Listing D.4: Python script used to prepare the data for extraction and compiling of V

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 18 14:24:42 2019

```

```
@author: akshay
"""

#Importing the required libraries
import numpy as np
import pandas as pd

#Read the ls.dat file which contains an ordered list of directory values
timeListing = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/ls.dat',header=None,sep='\t',dtype=str)

#User input for which wave gauge is to be analyzed
waveGaugeNumber = 6

#Read the data to store the y location of the selected wave gauge
location = np.loadtxt('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/undertow/60.1/WG'+str(waveGaugeNumber)+
'_U.xy',dtype=str,usecols=[0])

#Create a new empty dataframe which has y values as columns and add a time column
sampledData = pd.DataFrame(columns=[location])

#Adding the time column in the first location
sampledData.insert(0,'Time',None)

#Looping through all the time directories
for i in timeListing.iloc[:,0]:

    #Set the time value for the sampledData dataframe
    sampledData.loc[i,'Time'] = i

    #Create a dataLocation variable which changes as the time directory value changes
    dataLocation = '/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/undertow/'+str(i)+'/'

    #Read the data location file for a specified column only
    myData = pd.read_csv(str(dataLocation)+'WG'+str(waveGaugeNumber)+'_U.xy',header=None,sep='\t',usecols=[1,2])

    if (waveGaugeNumber == 6):
        lengthOfArray = 99
    else:
        lengthOfArray = 100

    #Looping through the 100 data values and storing the values for the given parameter
    for j in range(lengthOfArray):
        #Computing the velocity and saving it
        sampledData.loc[i,location[j]] = myData.iloc[j,0]
    #Print the current i index
    print("Time ",i," done!")

#Saving the data
sampledData.to_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/undertow_velocity_waveGauge_'
+str(waveGaugeNumber)+'.dat',sep='\t')
```

Listing D.5: Python script used to plot the results over a schematized flume

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 18 16:30:24 2019

@author: akshay
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


#What is to be plotted
plotVelocity = 1
saveFigure = 1

#Bounding values for the experimental flume
x1 = np.linspace(-5,0,50)
y1 = -0.4*np.ones(50)
x2 = np.linspace(0,15.95,125)
y2 = np.linspace(-0.4,-0.4+round(15.95/35,3),125)




#mean water level
mean_eta = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/meanSurfaceElevation_
waveFlow.txt',sep='\t',header=None)
xCoordOld = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/waveFlow/only_top.txt',sep='\t')
xCoord = xCoordOld.iloc[0][1:]-0.7

#Redefine xCoord for Standard deviation plot
xSD = xCoord.values

expMean = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/mean.csv',sep=',')

#Maximum water level
maximum_eta = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/maximum.dat',sep='\t',header=None)
maximum_eta_SD = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/maxSD.dat',sep='\t',header=None)
expMax = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/max.csv',sep=',')

#Minimum water level
minimum_eta = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/minimum.dat',sep='\t',header=None)
minimum_eta_SD = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/minSD.dat',sep='\t',header=None)
expMin = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/min.csv',sep=',')


#Loading the experimental velocity profiles
expWg2 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge2U.csv',sep=',',header=None)
```

```
expWg3 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge3U.csv',sep=',',header=None)
expWg4 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge4U.csv',sep=',',header=None)
expWg5 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge5U.csv',sep=',',header=None)
expWg6 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge6U.csv',sep=',',header=None)
expWg7 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge7U.csv',sep=',',header=None)
expWg8 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/expData/gauge8U.csv',sep=',',header=None)

#Loading experimental k profiles
expWg2k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge2.csv',sep=',',header=None)
expWg3k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge3.csv',sep=',',header=None)
expWg4k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge4.csv',sep=',',header=None)
expWg5k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge5.csv',sep=',',header=None)
expWg6k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge6.csv',sep=',',header=None)
expWg7k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge7.csv',sep=',',header=None)
expWg8k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/expData/kProfileGauge8.csv',sep=',',header=None)

#Loading the velocity profiles
wg2 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge2VelocityData.dat',sep='\t',header=None)
wg3 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge3VelocityData.dat',sep='\t',header=None)
wg4 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge4VelocityData.dat',sep='\t',header=None)
wg5 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge5VelocityData.dat',sep='\t',header=None)
wg6 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge6VelocityData.dat',sep='\t',header=None)
wg7 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge7VelocityData.dat',sep='\t',header=None)
wg8 = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY
/pythonCodeData/waveFlow/undertow/waveGauge8VelocityData.dat',sep='\t',header=None)

#Loading the k profiles
wg2k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge2kData.dat',sep='\t',header=None)
wg3k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge3kData.dat',sep='\t',header=None)
wg4k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge4kData.dat',sep='\t',header=None)
wg5k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge5kData.dat',sep='\t',header=None)
wg6k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge6kData.dat',sep='\t',header=None)
wg7k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge7kData.dat',sep='\t',header=None)
wg8k = pd.read_csv('/home/akshay/Desktop/Thesis/simulations/dataAnalysis/TINGandKIRBY/pythonCodeData/waveFlow/undertow/
waveGauge8kData.dat',sep='\t',header=None)

#Code for the flume
plt.figure(figsize=(15,9))
plt.xlim(4.5,10.5)
plt.ylim(-0.235,0.16)
plt.title("Ting and Kirby 1994 Comparison with waveFlow",fontsize=20)
plt.xlabel("X [m]",fontsize=20)
plt.ylabel("Y [m]",fontsize=20)
plt.fill_between(x1,-0.45,y1,color='grey')
plt.fill_between(x2,-0.45,y2,color='grey')

#Code for the mean eta
#plt.plot(xCoord[:-10],mean_eta[:-10],'-',color='black', label='$\overline{\eta}$')
#plt.plot(expMean['X'],expMean['Y'],'o', markersize=5, color='green')

#Plotting the SWL
plt.axhline(y=0,xmin=-2,xmax=12.5,linewidth=3,linestyle='--',color='darkgoldenrod')

#Plotting the zero velocity line for each wave gauge
plt.axvline(-1.265,linestyle='-.',color='black',alpha=0.4)
plt.axvline(5.945, linestyle='-.',color='black',alpha=0.4)
plt.axvline(6.665, linestyle='-.',color='black',alpha=0.4)
plt.axvline(7.275, linestyle='-.',color='black',alpha=0.4)
plt.axvline(7.885, linestyle='-.',color='black',alpha=0.4)
plt.axvline(8.495, linestyle='-.',color='black',alpha=0.4)
plt.axvline(9.11, linestyle='-.',color='black',alpha=0.4)
plt.axvline(9.725, linestyle='-.',color='black',alpha=0.4)

#Plotting the max and min profiles
plt.plot(xCoord[:-10],maximum_eta[:-10],'-',color='black',label='waveFlow')
plt.fill_between(xSD[:-10],maximum_eta.iloc[:-10,0]+maximum_eta_SD.iloc[:-10,0],
maximum_eta.iloc[:-10,0]-maximum_eta_SD.iloc[:-10,0],alpha=0.5,color='orange',label='1 S.D. Envelope')
#plt.plot(expMax['X'],expMax['Y'],'o', color='green', markersize=5)

#plt.plot(xCoord[:-10],minimum_eta[:-10],'-',color='black')
#plt.fill_between(xSD[:-10],minimum_eta.iloc[:-10,0]+minimum_eta_SD.iloc[:-10,0],minimum_eta.iloc[:-10,0]-minimum_eta_SD.iloc[:-10,0],alpha=0.5,color='orange')
plt.plot(expMin['X'],expMin['Y'],'o',color='crimson',markersize=7,label=r'TK94 ($\langle \eta_{max} \rangle$)')



#Plotting the velocity
if plotVelocity == 1:
    #Wave Gauge 2
    plt.plot(wg2[1][:np.where(wg2[0] > 0.5)[0][0]]+5.945,wg2[0][:np.where(wg2[0] > 0.5)[0][0]]*0.23,color='red', label='waveFlow (undertow)')
    plt.plot(expWg2[0]+5.945,expWg2[1]*0.23,'*',color='blue',label='TK94 (undertow)')

    #Wave Gauge 3
    plt.plot(wg3[1][:np.where(wg3[0] > 0.5)[0][0]]+6.665,wg3[0][:np.where(wg3[0] > 0.5)[0][0]]*0.2095,color='red')
    plt.plot(expWg3[0]+6.665,expWg3[1]*0.2095,'*',color='blue')

    #Wave Gauge 4
    plt.plot(wg4[1][:np.where(wg4[0] > 0.45)[0][0]]+7.275,wg4[0][:np.where(wg4[0] > 0.45)[0][0]]*0.1921,color='red')
    plt.plot(expWg4[0]+7.275,expWg4[1]*0.1921,'*',color='blue')

    #Wave Gauge 5
    plt.plot(wg5[1][:np.where(wg5[0] > 0.4)[0][0]]+7.885,wg5[0][:np.where(wg5[0] > 0.4)[0][0]]*0.1747,color='red')
    plt.plot(expWg5[0]+7.885,expWg5[1]*0.1747,'*',color='blue')

    #Wave Gauge 6
    plt.plot(wg6[1][:np.where(wg6[0] > 0.35)[0][0]]+8.495,wg6[0][:np.where(wg6[0] > 0.35)[0][0]]∗0.1573,color=’red’)
    plt.plot(expWg6[0]+8.495,expWg6[1]∗0.1573,’∗’,color=’blue’)

    #Wave Gauge 7
    plt.plot(wg7[1][:np.where(wg7[0] > 0.33)[0][0]]+9.11,wg7[0][:np.where(wg7[0] > 0.33)[0][0]]∗0.1397,color=’red’)
    plt.plot(expWg7[0]+9.11,expWg7[1]∗0.1397,’∗’,color=’blue’)

    #Wave Gauge 8
    plt.plot(wg8[1][:np.where(wg8[0] > 0.33)[0][0]]+9.725,wg8[0][:np.where(wg8[0] > 0.33)[0][0]]∗0.1221,color=’red’)
    plt.plot(expWg8[0]+9.725,expWg8[1]∗0.1221,’∗’,color=’blue’)

else:
#Plotting the k values
    plt.plot(wg2k[1][1:np.where(wg2k[0] > 0)[0][0]]+5.945,wg2k[0][1:np.where(wg2k[0] > 0)[0][0]]∗0.23,color=’red’,label=’waveFlow (TKE)’)
    plt.plot(expWg2k[0]+5.945,expWg2k[1]∗0.23,’∗’,color=’blue’,label=r’TK94 ($\langle k \rangle$)’)

    #Wave Gauge 3
    plt.plot(wg3k[1][1:np.where(wg3k[0] > 0)[0][0]]+6.665,wg3k[0][1:np.where(wg3k[0] > 0)[0][0]]∗0.2095,color=’red’)
    plt.plot(expWg3k[0]+6.665,expWg3k[1]∗0.2095,’∗’,color=’blue’)

    #Wave Gauge 4
    plt.plot(wg4k[1][1:np.where(wg4k[0] > 0)[0][0]]+7.275,wg4k[0][1:np.where(wg4k[0] > 0)[0][0]]∗0.1921,color=’red’)
    plt.plot(expWg4k[0]+7.275,expWg4k[1]∗0.1921,’∗’,color=’blue’)

    #Wave Gauge 5
    plt.plot(wg5k[1][1:np.where(wg5k[0] > 0)[0][0]]+7.885,wg5k[0][1:np.where(wg5k[0] > 0)[0][0]]∗0.1747,color=’red’)
    plt.plot(expWg5k[0]+7.885,expWg5k[1]∗0.1747,’∗’,color=’blue’)

    #Wave Gauge 6
    plt.plot(wg6k[1][1:np.where(wg6k[0] > 0)[0][0]]+8.495,wg6k[0][1:np.where(wg6k[0] > 0)[0][0]]∗0.1573,color=’red’)
    plt.plot(expWg6k[0]+8.495,expWg6k[1]∗0.1573,’∗’,color=’blue’)

    #Wave Gauge 7
    plt.plot(wg7k[1][1:np.where(wg7k[0] > 0)[0][0]]+9.11,wg7k[0][1:np.where(wg7k[0] > 0)[0][0]]∗0.1397,color=’red’)
    plt.plot(expWg7k[0]+9.11,expWg7k[1]∗0.1397,’∗’,color=’blue’)

    #Wave Gauge 8
    plt.plot(wg8k[1][1:np.where(wg8k[0] > 0)[0][0]]+9.725,wg8k[0][1:np.where(wg8k[0] > 0)[0][0]]∗0.1221,color=’red’)
    plt.plot(expWg8k[0]+9.725,expWg8k[1]∗0.1221,’∗’,color=’blue’)

plt.grid()
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

#Adding the breaking point line
plt.axvline(6.4, linestyle=’−−’,color=’darkmagenta’,label=’Breaking Point’,linewidth=3)

#Adding text in the plot
plt.text(5,0.001, ’MSL’,fontsize=15)

plt.legend(fontsize=13,loc=’upper left’)
#Save the figure
if saveFigure == 1:
    if plotVelocity == 1:
        plt.savefig(’flumeplotVelocity.png’,dpi=450)
    else:
        plt.savefig(’flumeplotK.png’,dpi=450)
plt.show()
```

Listing D.6: Python script used to plot the spectral results in a 3D plot

```
#!/usr/bin/env python3
# −∗− coding: utf−8 −∗−
"""
Created on Sat Jul 27 22:42:00 2019

@author: akshay
"""
#Importing the required libraries
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import PolyCollection
import matplotlib.pyplot as plt
from matplotlib import colors as mcolors
import numpy as np
import pandas as pd
import scipy.signal               #importing scipy.signal package for detrending
from scipy.fftpack import fft     #importing Fourier transform package
from scipy.stats import chi2      #importing confidence interval package

########################################################################################################################
#Function definition for the spectrum analysis (This code is taken from the course work by
Prof. Ad Reniers and Prof. Marion Tissier for the course work Ocean Waves at TU Delft)

def wave_spectrum(data,nfft,Fs):
    ’’’ Compute variance spectral density spectrum of the time−series and its
    90% confidence intervals.
    The time−series is first divided into blocks of length nfft before being
    Fourier−transformed.

    INPUT
      data      timeseries
      nfft      block length
       Fs       sampling frequency (Hz)

    OUTPUT
      E         variance spectral density. If data is in meters, E is in m^2/Hz
      f         frequency axis (Hz)
      confLow and confUpper      Lower and upper 90% confidence interval;
                                 (Multiplication factors for E)  ’’’

    # 1. PRELIMINARY CALCULATIONS
    # −−−−−−−−−−−−−−−−−−−−−−−−−−−−
    n = len(data)                  # length of the time−series
    nfft = int(nfft − (nfft%2))  # the length of the window should be an even number

    data = scipy.signal.detrend(data)      # detrend the time−series
    nBlocks = int(n/nfft)          # number of blocks (use of int to make it an integer)

    data_new = data[0:nBlocks*nfft] # (we work only with the blocks which are complete)

    # we organize the initial time-series into blocks of length nfft
    dataBlock = np.reshape(data_new,(nBlocks,nfft)) # each column of dataBlock is one block

    # 2. CALCULATION VARIANCE DENSITY SPECTRUM
    # ----------------------------------------

    # definition frequency axis
    df = Fs/nfft       # frequency resolution of the spectrum df = 1/[Duration of one block]
    f = np.arange(0,Fs/2+df,df)   # frequency axis (Fs/2 = Fnyquist = max frequency)
    fId = np.arange(0,len(f))

    # Calculate the variance for each block and for each frequency
    fft_data = fft(dataBlock,n = nfft,axis = 1)       # Fourier transform of the data
    fft_data = fft_data [:, fId]                  # Only one side needed
    A = 2.0/nfft*np.real(fft_data)                  # A(i,b) and B(i,b) contain the Fourier coefficients Ai and Bi for block b
    B = 2.0/nfft*np.imag(fft_data)                  #  -- see LH's book, page 325 for definition of Ai and Bi
                                                # /!\ assumes that mean(eta)=0

    E = (A**2 + B**2)/2                         # E(i,b) = ai^2/2 = variance at frequency fi for block b.

    # We finally average the variance over the blocks, and divide by df to get the variance DENSITY spectrum
    E = np.mean(E, axis = 0)/df

    # 3. CONFIDENCE INTERVALS
    # ------------------------
    edf = round(nBlocks*2) # Degrees of freedom
    alpha = 0.1                     # calculation of the 90% confidence interval

    confLow = edf/chi2.ppf(1-alpha/2,edf) # see explanations on confidence intervals given in lecture 3
    confUpper = edf/chi2.ppf(alpha/2,edf)

    return E,f,confLow,confUpper

########################################################################################################################

#Loading individual databases
#Importing the data
#data = pd.read_csv('waveFlow/waveGauges.dat',sep='\s+',skiprows=4,header=None)

#Importing the experimental data
expData = pd.read_csv('../expData/Rav5A.awg',sep='\s+',skiprows=1,header=None)
expTime = np.linspace(0,len(expData)*0.02,len(expData),endpoint=True)

#Importing OpenFOAM data
#foamData = pd.read_csv('waveFlow/postProcessing/surfaceElevationAnyName/0/surfaceElevation.dat', sep='\s+',skiprows=4,header=None)

#Define the shift parameters
#timeDisp = [-0.1288,-1.99195,-0.47595,-0.371,0,0,0,1.341,0]
#move = [-0.0013,-0.0003,-0.0006,-0.0007,0,0,0,-0.002,0]


#Define the surface plot for the flume
flumeY = np.linspace(0,46,200)
flumeZ = np.zeros(200)

#Index and assign Y coordinate for the flume
flumeZ[:48] = 0 #First 11 meters is y=0
flumeZ[48:71] = np.linspace (0,0.62,23, endpoint=True) #The next 5 meters linearly increase
flumeZ[71:] = np.linspace (0.62,0.95,129, endpoint=True) #The rest of the domain linearly increases



#Index and assign the values



#EndTime definition
endTime = 110

#Define cut-off frequency
cutOffFrequency = 1.5




#Computing the spectrum for the given wave paddle series
WG1 = wave_spectrum(expData[1],len(expData[1])*0.01,50) #Absolute Location 3.415 m
WG7 = wave_spectrum(expData[6],len(expData[6])*0.01,50) #Absolute Location 14.84 m
WG26 = wave_spectrum(expData[9],len(expData[9])*0.01,50) #Absolute Location 45.79 m
WG27 = wave_spectrum(expData[10],len(expData[10])*0.01,50) #Absolute Location 45.96 m

#Specify the X axis and slice it up until when the cut-off frequency is 1.5
xs1 = WG1[1][:np.where(WG1[1] > cutOffFrequency)[0][0]]
xs7 = WG7[1][:np.where(WG7[1] > cutOffFrequency)[0][0]]
xs26 = WG26[1][:np.where(WG26[1] > cutOffFrequency)[0][0]]
xs27 = WG27[1][:np.where(WG27[1] > cutOffFrequency)[0][0]]

#Specify the Y axis
ys1 = (WG1[0][:np.where(WG1[1] > cutOffFrequency)[0][0]])/max(WG1[0])
ys7 = (WG7[0][:np.where(WG7[1] > cutOffFrequency)[0][0]])/max(WG7[0])
ys26 = (WG26[0][:np.where(WG26[1] > cutOffFrequency)[0][0]])/max(WG26[0])
ys27 = (WG27[0][:np.where(WG27[1] > cutOffFrequency)[0][0]])/max(WG27[0])

#Starting and ending points set to zero for plot consistency
ys1[0], ys1[-1] = 0,0
ys7[0], ys7[-1] = 0,0
ys26[0], ys26[-1] = 0,0
ys27[0], ys27[-1] = 0,0

#Specify the Z axis
zs = [3.415, 14.84, 42.79, 45.96] #Location of the wave gauges

#Create and empty sotrage for verts
verts = []


#Define what verts are
verts.append(list(zip(xs1,ys1)))
verts.append(list(zip(xs7,ys7)))
verts.append(list(zip(xs26,ys26)))
verts.append(list(zip(xs27,ys27)))

#Define how to assign the colors
def cc(arg):
    return mcolors.to_rgba(arg, alpha=0.6)


##Setting the figure definition
#fig = plt.figure(figsize=(20,15))
#ax = fig.gca(projection='3d')



fig = plt.figure(figsize=(20,15))
ax = fig.gca(projection='3d')
#Plotting the 3D spectrum plots
poly = PolyCollection(verts,facecolors=[cc('r'),cc('g'),cc('b'),cc('olive')], edgecolor='black')
ax.add_collection3d(poly, zs=zs, zdir='y')
ax.plot(flumeY,flumeZ,zs=1.5,zdir='x',color='black',linewidth=2)

#for i in np.linspace(0,360,361):
ax.view_init(elev=10,azim=125)

#Formatting
ax.set_xlabel(r'Frequency [$Hz$]',fontsize=15,labelpad=15)
ax.set_xlim3d(0, 1.5)
ax.set_ylabel(r'Flume Location [m]',fontsize=15,labelpad=15)
ax.set_ylim3d(0, 46)
ax.set_zlabel(r'Normalized Variance Density [$-$]',fontsize=15,labelpad=15)
ax.set_zlim3d(0, 1)
#plt.savefig('Results/animation/spectrum/spectrum_frame.png',dpi=300)
#print(i)
```

Listing D.7: Python script used to carryout the zero crossing analysis

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 7 00:08:01 2019

@author: akshay
"""
import numpy as np

#Coding the definition to detect the zero crossing
def zeroCrossing(data,gauge):
    ''' This function detects the zero crossing by saving the indices where the zero value is reached

        data: is the data stream which needs to be analyzed
        gauge: is the gauge number within the data file

        RETURNS:
            this function returns a list of all the indices where zero crossing occurs
    '''
    #Empty list for array indices
    ones = []

    #Loop through the dataset and locate where there is a zero crossing
    for i in range(0,len(data.iloc[:, gauge])-1):
        #Check using the zero-up crossing
        if (data.iloc[i,gauge] <= 0) and (data.iloc[i+1,gauge] > 0):
            ones.append(i)

    return ones


def minMaxAnalysis(givenData,resultsArray,gauge):
    '''
        This function computes the minimum and maximum by windowning the given time series based on zero up crossing

        INPUT:

        givenData = Original dataset
        resultsArray = Result obtained from the zeroCrossing function
        gauge = Gauge number in givenData

        OUTPUT:

        The function return the following data

        Data            Slicing Index
        maxWaveHeight   - 0
        minWaveHeight   - 1
        maxIndex        - 2
        minIndex        - 3
    '''
    #Compute and store the minimum and maximum values
    maxWaveHeight = []
    minWaveHeight = []
    maxIndex = []
    minIndex = []

    #Loop through the indices and compute the max and minimum indices
    for i in range(0,len(resultsArray)-1):
        #Computing the max within the given window
        maxWaveHeight.append(max(givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge]))

        #Storing the location where this max exists within this window
        maxIndex.append(resultsArray[i] + np.where(max(givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge]) == givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge])[0][0])

        minWaveHeight.append(min(givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge]))

        #Storing the location where this max exists within this window
        minIndex.append(resultsArray[i] + np.where(min(givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge]) == givenData.iloc[resultsArray[i]:resultsArray[i+1],gauge])[0][0])

    return maxWaveHeight, minWaveHeight, maxIndex, minIndex
```

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.