MODELLING OF A WAVE FLUME IN OPENFOAM

By

Marmol Dimitri Paul

/GC6GXJ/

Submitted to the
Department of Fluid Mechanics of the
Budapest University of Technology and Economics
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering Modelling

On the 28th of December, 2020

MSc Thesis

Final project B/ BMEGEÁTNKDB

Supervisor:

Josh Davidson

Department of Fluid Mechanics
Faculty of Mechanical Engineering
Budapest University of Technology and Economics

Declaration

Full Name (as in ID):	Marmol Dimitri Paul
Neptun Code:	GC6GXJ
University:	Budapest University of Technology and Economics
Faculty:	Faculty of Mechanical Engineering
Department:	Department of Fluid Mechanics
Major/Minor:	MSc in Mechanical Engineering Modelling Fluid Mechanics major / Solid Mechanics minor
Thesis Title:	Modelling of a wave flume in OpenFoam
Academic year of submission:	2020-2021-I.

I, the undersigned, hereby declare that the Thesis submitted for assessment and defence, exclusively contains the results of my own work assisted by my supervisor. Further to it, it is also stated that all other results taken from the technical literature or other sources are clearly identified and referred to according to copyright (footnotes/references are chapter and verse and placed appropriately). I accept that the scientific results presented in my Thesis can be utilised by the Department of the supervisor for further research or teaching purposes.

Budapest, 28th of December, 2020

Acknowledgments:

I would like to thank my supervisor Josh Davidson, who, with patience, guided me through this entire work. Even in a global crisis as we are experiencing nowadays, we could continue this study into its end.

Thanks to my family and friends for their support.

Abstract

In this document, experimental results from previous work were analyzed then used in order to model a waveflume in OpenFoam. The theories are firstly studied to extract all the essential parameters that are necessary to justify the good use of the numeric model. A specific case was used in the intermediate water waves field. It was found that, despite the use of the laminar type of simulation, the extrapolation method and the Wheeler stretching method are not describing well the results obtained by the simulation. Further work on different type of simulations must be made to confirm or invalidate the previous assessment.

Table of content

Nomenclature:

- **CFD** : Computational Fluid Dynamics
- **WEC** : Wave Energy Converter
- **NWT** : Numerical Wave Tank
- **t** : time (s)
- **ρ** : density ($kg.m^{-3}$)
- **u** : velocity vector ($m.s^{-1}$)
- **p** : pressure (Pa)
- $\mathbf{f_b}$: body force (N)
- **e** : energy per unit mass ($J.kg^{-1}$)
- $\mathbf{q_s}$: rate of heat transfer per unit area [$W.m^{-2}$]
- $\mathbf{q_v}$: is the rate of heat sink per unit area [$W.m^{-2}$]
- **τ :** is the stress tensor [Pa]
- **ν** is the kinematic viscosity [m^2/s]
- **FSE** :Free Surface Elevation
- **η** : FSE [m]
- **H**: the height [m]
- **λ**: the wavelength [m]
- **T** : the time period [s]
- **d**: the water depth [m]
- **a**: the wave amplitude [m]
- **F**: the frequency of the wave [Hz]
- **ω** : the angular frequency [rad/s]
- **k**: the wave number [m^{-1}]
- **s :** the wave steepness
- **PIV :** Particule Image Velocimetry
- **VOF** :Volume of Fluid method
- **FVM** :Finite Volume Method
- α : volume fraction

List of figures:

1 Introduction

The growth of the earth's population is intrinsically linked to the access of energy. Fossil energies are massively used due to the ease of extracting them and converting them. However because of this intense use of fossil energies, the environment is getting polluted. In the wake of a massive extinction, we need to find environmentally friendly solutions to get energy from renewable sources. Among them, wave energy has a high potential. Fed by the wind, which is fueled from the sun, the density of power of the waves is of the order of 40-60 kW/m as it is shown in the figure below:

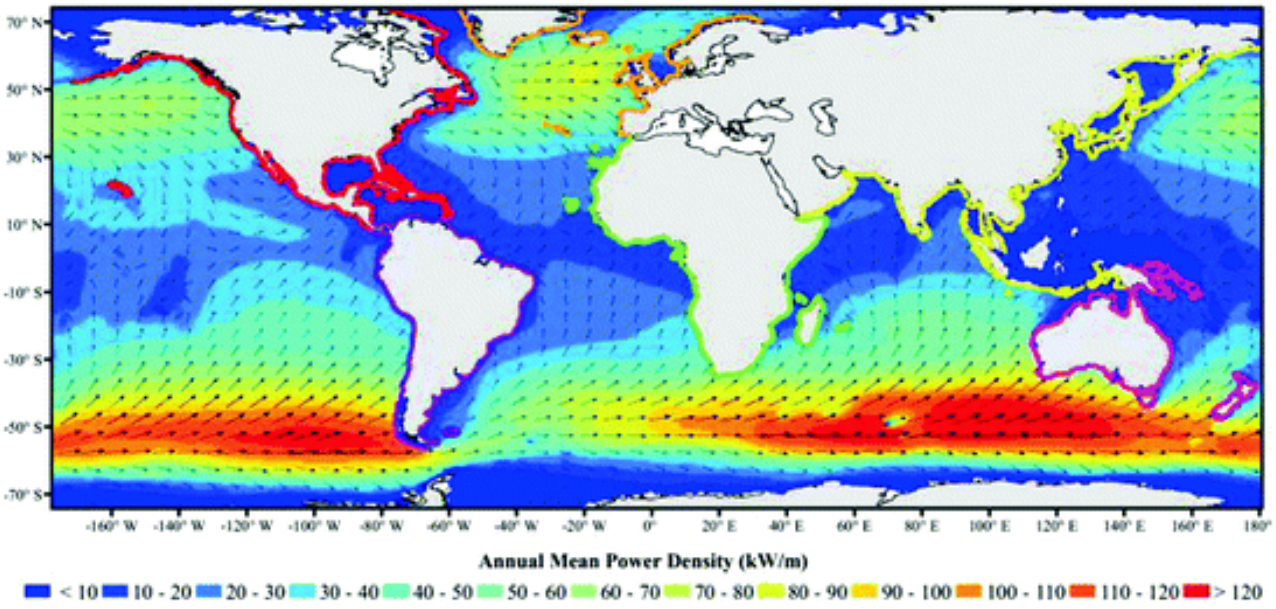


Figure 1 The global wave energy resource [18]

In order to extract the energy from the waves, many Wave Energy Converters (WECs) are currently under studies, but to optimize the energy conversion, it is crucial to understand the wave kinematics. Therefore a set of experience was conducted in the university of Plymouth, UK, in order to validate wave theories such as Wheeler stretching.

High resources are needed in order to carry out a real experiment. Therefore it is now common to use a numerical method to model the reality. Indeed numerical models provide high resolution solutions, repeatability. Moreover real

experiments are subject to errors due to, for instance, measurement errors, mechanical errors. Numerical models use wave theories in order establish a solution, but some theories are not fully validated. Thus both real and numerical models have to be compared in order to validate theories. For this study OpenFoam is used because it is a free opensource software that requires less resources than for instance Fluent.

The aim of this project will be to develop a Numerical Wave Tank (NWT) replication of the experiments carried out in Plymouth, to provide validation of the modelling of a wave flume in OpenFOAM. This paper will compare the results of the real experience and the NWT by looking at the different velocity profiles and the free surface elevation. The first part will tackle the issue of explaining which wave theories will be used and which will be under study as well as an overview of previous studies made on the NWT. Then the NWT will be entirely explained, indeed many conditions must be fulfilled in order to use the results of the simulation on the waves. Finally, the results will be compared from the real experiment and the simulation.

2 Mathematical formulation of the physical problem

2.1 Governing equations

2.1.1 Basic equations

The equations leading to the understanding of the phenomenon rely on several law of physics [1] :

- Conservation of the mass, which states that the mass of a closed system remains constant over time :

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$$

Where ρ is the density, **u** is the velocity field.

In particular in our case, we can consider that the fluid is incompressible. The equation becomes then

$$\nabla \cdot \boldsymbol{u} = 0$$

- The conservation of the linear momentum (2nd law of Newton). In particular, for our case of study, the equation that's results of this law is

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \boldsymbol{f_b}$$

Where $\mathbf{f_b}$ is the body force.

- The first law of thermodynamics, which states that the energy of an isolated system remains constant

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\rho \boldsymbol{u} e) = -\nabla \cdot q_s - \nabla \cdot \rho \boldsymbol{u} + \nabla(\boldsymbol{\tau} \cdot \boldsymbol{u}) + \boldsymbol{f_b} \cdot \boldsymbol{u} + q_v$$

Where e is the energy per unit mass, q_s is the rate of heat transfer per unit area across the area of the material element, q_v is the rate of heat sink or source transfer within the material volume per unit volume and $\boldsymbol{\tau}$ is the stress tensor.

2.1.2 Navier-Stokes equations

The Navier-Stokes equations express mathematically the conservation of momentum and the conservation of the mass for Newtonian fluids. It has been written as follows:

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla \boldsymbol{p} + \nabla \boldsymbol{\tau} + \rho \boldsymbol{f}$$

Where :

- t is time [s]
- ρ is the density of the mixture [kg.m^{-3}]
- **u** is the velocity field [m.s^{-1}]
- p is the pressure [Pa]
- τ is the deviatoric stress tensor, also known as Cauchy stress tensor [N.m^{-2}]
- **f** represents body accelerations on the continuum

For this study, we consider that **f=g** where the only acceleration considered is the gravity.

For incompressible fluid, the equation becomes

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu = -\frac{\nabla \boldsymbol{p}}{\rho} + \boldsymbol{g}$$

Where ν is the kinematic viscosity.

2.1.3 Wave characterization

A basic wave can be characterized by a sinusoidal variation of the Free Surface Elevation (FSE). Therefore this basic wave can be defined by some variables as shown in the picture below:

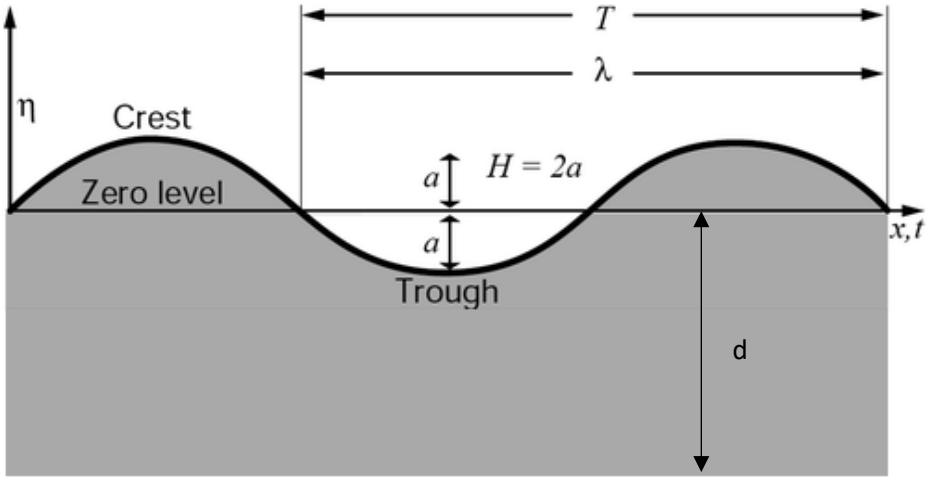


Figure 2 Definition of wave parameters over a basic wave

We can list the characteristics of the wave:

- **H** *[m]* : the height, which is the distance between the crest and the trough of the wave
- **λ** *[m]* : the wavelength, which represents the distance between two identical spatial points of the wave
- **T** *[s]* : the time period, which is the time for the wave to repeat.
- **d** *[m]* : the water depth
- **a** *[m]* : the wave amplitude

Besides, to fully characterize a wave, we need to calculate those different parameters:

- **F** *[Hz]* : the frequency of the wave $F = \frac{1}{T}$
- **ω** *[rad/s]* : the angular frequency $\omega = \frac{2\pi}{T}$
- **k** *[m^{-1}]* : the wave number $k = \frac{2\pi}{\lambda}$

- **s :** the wave steepness $s = k * a$

The Le Mehauté diagram, shown below, illustrates and resumes the wave theories which are used now. The wave theories provide an analytical expression of the velocity profile for a constant water depth.

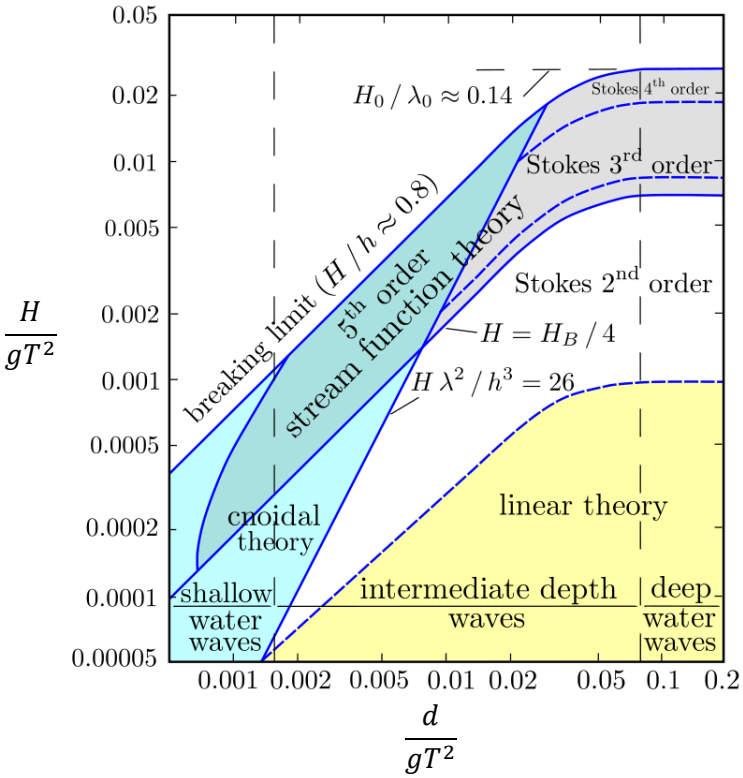


Figure 3 Le Méhauté diagram [19]

This diagram helps us to choose a theory, by looking at the wave steepness s, the wave amplitude H, the wave frequency f and the water depth.

The free surface elevation and the velocity profile is well known in the linear theory and is the simplest theory. Indeed the FSE is given by:

$$\eta^{(1)} = \frac{H}{2} * \cos(\theta)$$

Where θ is the phase function $kx - \omega t$, x is the direction of the wave propagation. The velocity profile underneath the free surface elevation was summarized in [17]. The formulas for the horizontal and the vertical velocities are given by:

$$u^{(1)} = \frac{\pi H}{T}\frac{\cosh\left(k(z+d)\right)}{\sinh\left(kd\right)}\cos\left(\theta\right)$$

$$v^{(1)} = \frac{\pi H}{T}\frac{\sinh\left(k(z+d)\right)}{\sinh\left(kd\right)}\sin\left(\theta\right)$$

By increasing the wave height and the steepness, we must extend the description of the free surface elevation to second-order Stokes waves with the following terms:

$$\eta^{(2)} = \eta^{(1)} + \frac{\pi H^2}{8\lambda}\frac{\cosh(kd)}{sinh^3(kd)}(2+\cosh(2kd))\cos\left(2\theta\right)$$

Thus the horizontal and vertical velocities are given by :

$$u^{(2)} = u^{(1)} + \frac{3}{4}\frac{\pi H}{T}\left(\frac{\pi H}{\lambda}\right)\frac{\cosh\left[2k(z+d)\right]}{sinh^4(kd)}\cos\left(2\theta\right)$$

$$v^{(2)} = v^{(1)} + \frac{3}{4}\frac{\pi H}{T}\left(\frac{\pi H}{\lambda}\right)\frac{\sinh\left[2k(z+d)\right]}{sinh^4(kd)}\sin\left(2\theta\right)$$

These formulas were originally developed for deep water waves. Therefore, they might be not well suited for intermediate conditions. In addition, the velocities formulas above are calculated up to the still water level, so additional analytical descriptions, such as extrapolation or stretching, have to be used to describe the in-crest velocities.

2.1.4 Extrapolation and Wheeler stretching

The first approach to this issue is to use a linear extrapolation within the crest. The sill water level is noted z = 0. The horizontal and the vertical velocities follow then this rule: for z ≤ 0, u and v are calculated with the appropriate wave theory and for z > 0, u and v are given by :

$$u(z) = u(0) + z\left.\frac{du}{dz}\right|_{z=0}$$

$$v(z) = v(0) + z\left.\frac{dv}{dz}\right|_{z=0}$$

Another method is to use the Wheeler stretching [17]. A new vertical coordinate, $z_s(t)$, is defined from the sea floor to the free surface, $-d < z_s(t) < \eta(t)$. This new parameter is related to the z coordinate lying between $-d < z < 0$, which is then substituted in the equation:

$$z = \frac{z_s - \eta}{1 + \frac{\eta}{d}}$$

While the extrapolation method does not alter the velocity profiles below the still water level, Wheeler stretching implies changes in the velocity profiles below z = 0. The influence of the extrapolation (solid lines) and Wheeler stretching methods (dashed lines) is illustrated in figures 4 and 5, showing the horizontal velocity profiles along the water depth, under the wave crest and trough. The waves are clustered into two groups: intermediate water waves (from I1 to I5) and deep-water waves (from D1 to D5). Within each group, the individual waves have the same period and wavelength, and increasing wave heights. From Figures 4 and 5, it can be seen that the profiles based on Wheeler stretching method have a larger radius of curvature than the profile based on extrapolation method under crests. The opposite trend is observed under troughs. So the two methods deliver different results, with an increasing discrepancy with increasing wave height. The accuracy of each method was assessed based on the experimental results according to this work [7]. Also, the figure 4 show the effect of intermediate water conditions: the velocity at the bottom of the tank is not close to zero in the theoretical profiles, even though it is zero in the physical environment due to the friction with the bottom wall. This implies a small curvature in the theoretical profiles from the sea floor to the free surface which may not represent the reality.

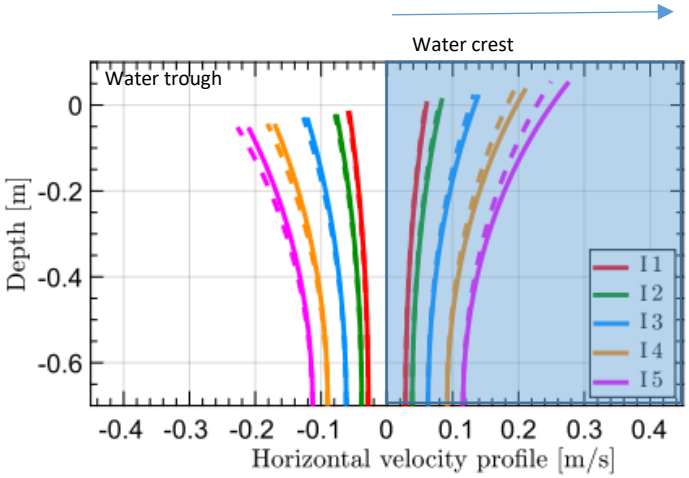


Figure 5 Velocity profile for intermediate water conditions [20]

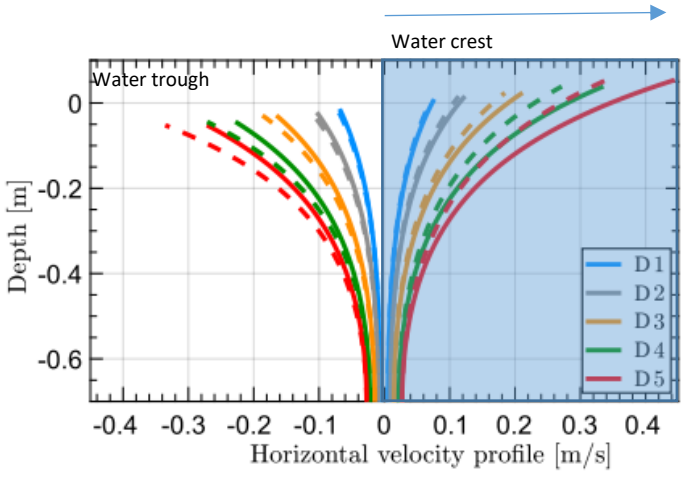


Figure 4 Velocity profile for deep water conditions [20]

3 OpenFOAM

This section is intended to give an overview of the numerical methods implemented in OpenFoam through the description of the solver used for this work. We will see through this section, how the free surface is modelled, and which equations are added to the set of equations that must originally be solved. Then we will see how with the finite volume method, these equations are discretized into the cells of the mesh. The solver interFoam, which has been used originally for this study, will be then detailed. More information can be found in the OpenFoam user guide [8]

3.1 General overview

OpenFOAM is a C++ library, used to create applications. The applications fall into two categories: solvers, which are designed to solve the problem in continuum mechanics; utilities, which are designed to manipulate data. Since OpenFOAM uses a C++ language, extensive modifications can be implemented by the users as the creation of solvers, which will be discussed later in this paper with interFoamBeach. The overall structure of OpenFOAM is shown is shown in the figure below:

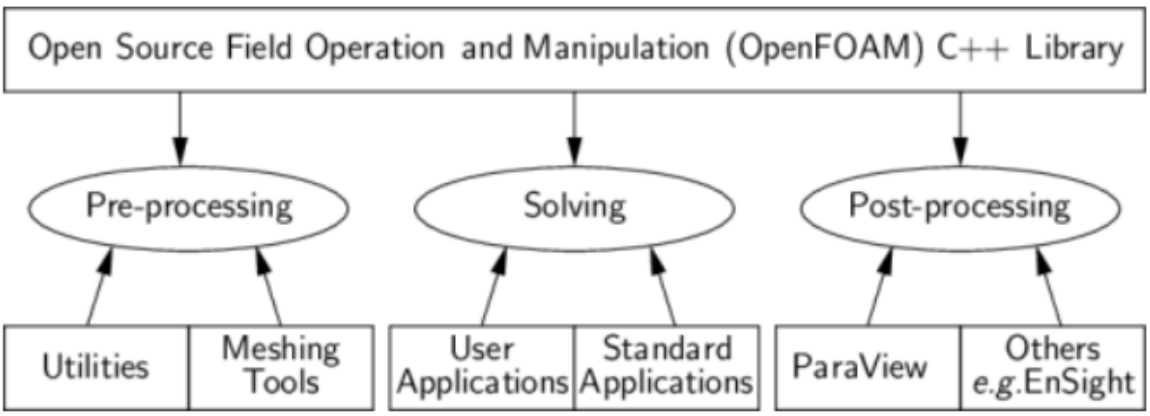


Figure 6 Overview of OpenFOAM structure [8]

Compared to other commercial software like ANSYS FLUENT, on OpenFOAM, the simulation and its options are entirely defined on text files. These text files are sorted as shown in the figure below:

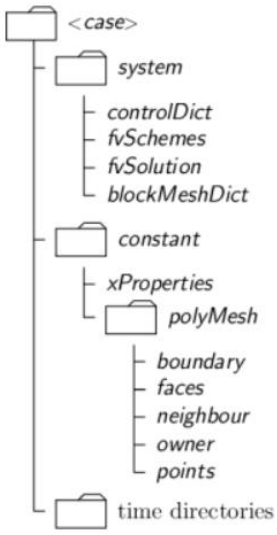


Figure 7 Case directory structure [8]

- **system:** Contains the geometry and mesh definition, numerical scheme settings and the simulation controls.

- **constant:** Contains the physical properties of the case. Any boundary or interface modifications during simulation is set up in this folder with their respective "dictionary" file.

- **0.org:** Contains information for physical values, volume fractions, and boundary conditions. This is the start folder for the simulation that copied to be 0 folder for the first-time step.

With this structure, OpenFOAM has many advantages and disadvantages:

OpenFOAM is released under the license GNU General Public License, which means that it is free for anyone to download and use. OpenFOAM also allows the possibility to use all available processors for a single simulation, whereas for commercial software licenses, the use of additional processors requires additional licenses, limiting therefore the available processors to the number purchased in the license. Moreover, the lack of license constraints means that OpenFOAM can be customized to suit any workflow. Tasks ordinarily requiring manual interaction can be automated using the python library PyFoam.

However, OpenFOAM is not controlled through a graphical user interface (GUI) unlike most of the commercial CFD softwares. Settings are adjusted via text files called dictionaries and everything is controlled via the command line. The lack of a GUI and of a maintained documentation makes it difficult for the new users.

For post-processing simulations OpenFOAM comes with ParaView that allows the visualization of the geometry, the results and some post processing manipulations. It is called with paraFoam command from the case folder.

3.2 Free surface boundary condition

Free surfaces occur at the interface between two fluids. Such interfaces require two boundary conditions to be applied: a kinematic condition which relates the motion of the free interface to the fluid velocities at the free surface and a dynamic condition which is concerned with the force balance at the free surface.

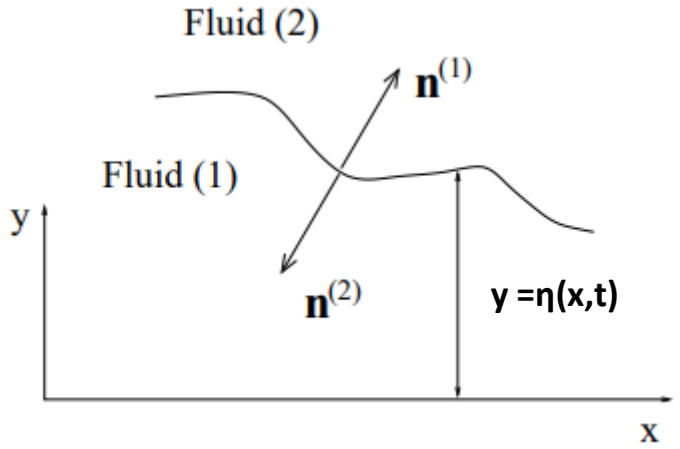


Figure 8 Free surface boundary condition

3.2.1 Kinematic boundary condition

The position of a free surface can always be given in implicit form as $F(x_j , t) = 0$. For instance, the height of the free surface above the x-axis is specified as

$y = \eta(x, t)$ and an appropriate function $F(x, y, t)$ would be given by $F(x, y, t) = \eta(x, t) - y$.

Fluid particles on the free surface always remain part of the free surface, therefore we must have

$$\frac{DF}{Dt} = \frac{\partial F}{\partial t} + u_k \frac{\partial F}{\partial x_k} = 0$$

3.2.2 Dynamic boundary condition

The dynamic boundary condition requires the stress to be continuous across the free surface which separates the two fluids (air and water in this study case). The traction exerted by fluid (1) onto fluid (2) is equal and opposite to the traction exerted by fluid (2) on fluid (1). Therefore we must have $t^{(1)} = -t^{(2)}$. Since $n^{(1)} = -n^{(2)}$. By neglecting the free surface tension, we obtain

$$\tau_{ij}^{(1)} n_j = \tau_{ij}^{(2)} n_j$$

3.3 Volume of fluid method

Several methods were developed to tackle the question of multiphase flows modelling considering the free surface boundary issue. For our study case, a well-accepted method will be used: the Volume of Fluid (VOF) method.

The VOF method is reported by Nichols and Hirt in their papers [2][3] and consists of three ingredients: a scheme to locate the surface, an algorithm to track the surface as a sharp interface moving through a computational grid, and a means of applying boundary conditions at the surface. Therefore the density and viscosity are weight-averaged by the volume fraction of each phase:

$$\rho = \alpha_l \rho_l + (1 - \alpha_l) \rho_g$$

$$\mu = \alpha_l \mu_l + (1 - \alpha_l) \mu_g$$

where α_l is a volume fraction of liquid, α_g volume fraction of gas and ρ_l and ρ_g are densities of liquid and gas respectively. Velocity also determined on weighted-average method as

$$u = \frac{\alpha_l \rho_l u_l + \alpha_g \rho_g u_g}{\rho}$$

This method allows the treatment of each phase separately. The evolution of the fluid in the system is governed by the following transport equation:

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l \boldsymbol{u}) = 0$$

With the following constraints, since there are only 2 phases in the system:

$$\alpha_l + \alpha_g = 1$$

The VOF method is computationally friendly, as it introduces only one additional equation and thus requires minimal storage.

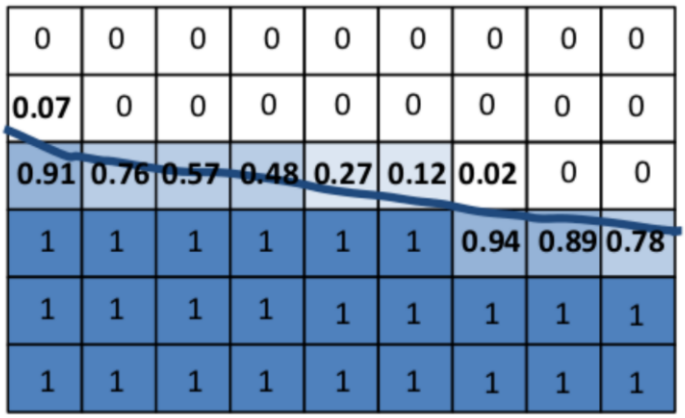


Figure 9 VOF method depicting the free surface [21]

3.4 Finite volume method

The Finite Volume Method (FVM) is a discretization method which is well suited for the numerical simulation of various types of conservation laws; it has been extensively used in several engineering fields, such as fluid mechanics. It may be used on arbitrary geometries, using structured or unstructured meshes, and it leads to robust schemes. An additional feature is the local conservativity of the numerical fluxes, that is the numerical flux is conserved from one discretization cell to its neighbour. This last feature makes the finite volume method quite attractive when modelling problems for which the flux is of importance, such as in fluid mechanics. The finite volume method is locally conservative because it is based on a "balance" approach: a local balance is written on each discretization cell which is often called "control volume"; by the divergence formula, an integral formulation of the fluxes over the boundary of the control volume is then obtained. The fluxes on the boundary are discretized with respect to the discrete unknowns. More information can be found in this paper [4].

3.4.1 Explanations of FVM

The basis of the finite volume method is the integral conservation law. The essential idea is to divide the domain into many control volumes (or cells) and approximate the integral conservation law on each of the control volumes. The figure below shows an example of a partition of a one-dimensional domain into cells. By convention cell i lies between the points $\boldsymbol{x}_{i-\frac{1}{2}}$ and $\boldsymbol{x}_{i+\frac{1}{2}}$.

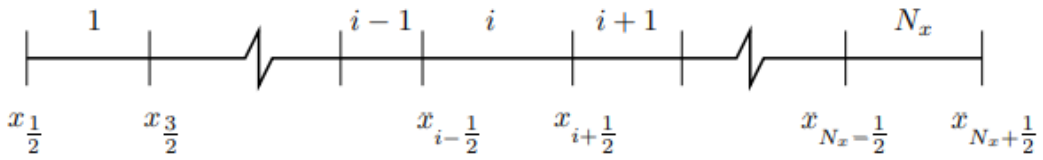


Figure 10 Mesh and notation for one-dimensional finite volume method [22]

We recall the general integral conservation law :

$$\frac{d}{dt}\int_{(\Omega)} U d\Omega + \int_{d\Omega} F(U)\cdot n ds = \int_{(\Omega)} S(U,t) d\Omega$$

With

- Ω represents the domain
- ∂Ω represents the boundary of the domain
- U(x, y, z, t) represents a physical quantity, it is a scalar, integrating over the domain it will be dependent only on time
- F is the Flux vector that represents inflow and/or outflow on the domain via the boundaries, n is the outward pointing normal vector of the boundary
- S represents volume sources that can be either sources or sinks.

By using the Green-Ostrogradsky theorem, we can therefore write that

$$\int_{d\Omega} F(U)\cdot n ds = \int_{(\Omega)} \nabla\cdot F \, d\Omega$$

So the equation can be rewritten as :

$$\frac{d}{dt}\int_{(\Omega)} U d\Omega + \int_{(\Omega)} \nabla \cdot F \, d\Omega = \int_{(\Omega)} S(U,t) d\Omega$$

This form is called the integrated form, it is considered as a global form. We can also write a local form, a differential form :

$$\frac{dU}{dt} + \nabla \cdot F(U) = S(U,t)$$

3.4.2 Transport equation

The Navier-Stokes equation can be written using the standard form of the transport equation for any quantity $\Phi(t)$:

$$\frac{\partial \rho \Phi(t)}{\partial t} + \nabla \cdot \left(\rho \boldsymbol{U} \Phi(t)\right) = \nabla \cdot \left(\Gamma \nabla \Phi(t)\right) + S_{\Phi}$$

Where

- ρ is the density, U is the velocity and Γ is the diffusivity.
- $\frac{\partial \rho \Phi(t)}{\partial t}$ is the transient term, which represents the accumulation of $\Phi(t)$ in the concerned control volume.
- $\nabla \cdot \left(\rho \boldsymbol{U} \Phi(t)\right)$ is the convection term, which represents the transport of $\Phi(t)$ due to the existence of a velocity field.
- $\nabla \cdot \left(\Gamma \nabla \Phi(t)\right)$ is the diffusion term which represents the transport of $\Phi(t)$ due too its gradient
- S_{Φ} is the source term, which represents any sources or sinks that create or destroy $\Phi(t)$.

3.4.3 Transient term

The integral form of the transient term is :

$$\int_{V} \frac{\partial \rho \Phi(t)}{\partial t} dV$$

The discretization follows the next rule:

- $\Phi^{(n)} = \Phi(t + \Delta t)$ the new values at the solving time step

- $\Phi^{(n-1)} = \Phi(t)$ the previous values which are stored from the previous time step
- $\Phi^{(n-2)} = \Phi(t - \Delta t)$ older values

The discretization method can be implemented in the fvSchemes file under the ddtSchemes section. For this study, we use the Euler scheme. More information can be found in the user guide [5]

3.4.4 Convection term

The convection term is integrated and linearized as follows:

$$\int_V \nabla \cdot \left(\rho \boldsymbol{U} \Phi(t)\right) dV = \sum_f F \Phi_f$$

Where F is the mass flux through the face f.

If we consider the momentum equations, φ is replaced by the velocity vector which is expressed at the cell centers. Therefore an interpolation of the values from cell centers to face centers is needed. A detailed method can be found in this help note to choose [14].

These parameters are written in the fvSchemes file, under the divSchemes section

```
divSchemes
{
    div(rhoPhi,U)   Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss interfaceCompression;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((muEff*dev(T(grad(U))))) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
```

Figure 11 Selected schemes for this study

3.4.5 Diffusion term

As for the convection term, the diffusion term is also integrated and linearized as follows:

$$\int_V \nabla \cdot (\Gamma \nabla \Phi) dV = \sum_f \Gamma_f S_f \cdot (\nabla \Phi)_f$$

Where Γ is considered as a scalar.

This parameter is also written in the fvSchemes files under the section Laplacian. In order to transform cell-centre quantities to face centres, an interpolation scheme has to be used. The linear interpolation was used in this study.

```
laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}
```

Figure 12 Laplacian schemes

4 Numerical study

4.1 Real tank setup

During the real experiment, the tank was set up like in the picture below:

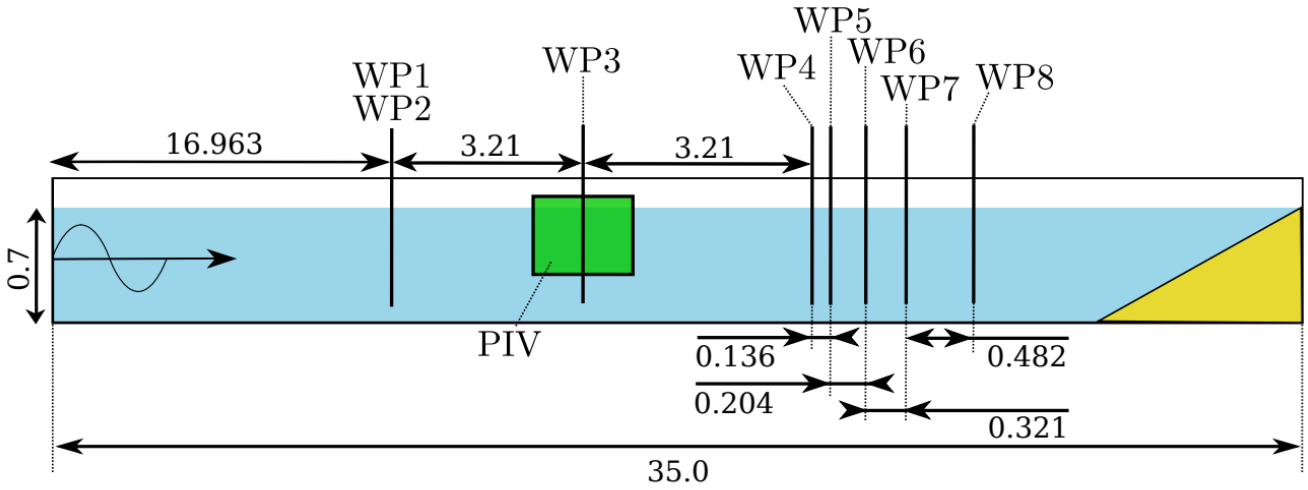


Figure 13 Tank setup [7]

A piston on the left of the tank generated a wave and the Particle Image Velocimetry (PIV) was employed in order to define the velocity profile underneath the wave. The Wave Probes (WP) 1-2 are located in the same horizontal distance from the piston, the WP 3 is located in the center of the interrogation window and the WP 4-8 are located before the beach in order to calculate the refraction coefficient.

10 experiments were carried out, where 10 different waves were created. Because the linear case is already well studied in [17], the focus was on the second and third order of Stokes and in an intermediate-water or deep-water conditions. The waves from I1 to I5 are intermediate-water waves with a wave period of 1.53s and a wavelength of 3.21m. The waves from D1 to D5 are deep-water waves with

a wave period of 0.94s and a wavelength of 1.36m. All the cases are detailed in the figure and the chart below.

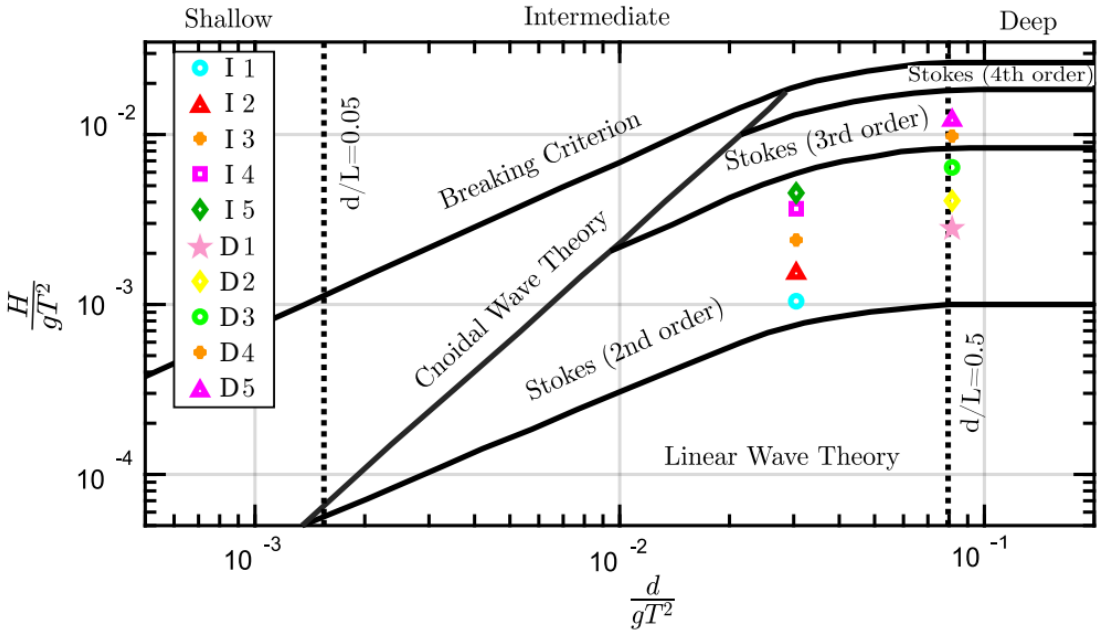


Figure 14 Experience range [7]

Case ID	T [s]	H [m]	λ [m]	H/λ [–]	d/λ [–]
I1	1.53	0.025	3.21	0.0078	0.22
I2	1.53	0.034	3.21	0.0106	0.22
I3	1.53	0.055	3.21	0.0171	0.22
I4	1.53	0.080	3.21	0.0249	0.22
I5	1.53	0.101	3.21	0.0315	0.22
D1	0.94	0.021	1.36	0.0154	0.51
D2	0.94	0.032	1.36	0.0235	0.51
D3	0.94	0.054	1.36	0.0397	0.51
D4	0.94	0.085	1.36	0.0625	0.51
D5	0.94	0.112	1.36	0.0824	0.51

Figure 15 Details of the experimental range, T for wave period, H for wave height, λ for wavelength and d for depth [7]

In the end of the tank, a wave-absorbing beach was constructed in a wave-absorbing foam in order to have the lowest coefficient of reflection possible.

Shown below, pictures and schematics of the experiments:

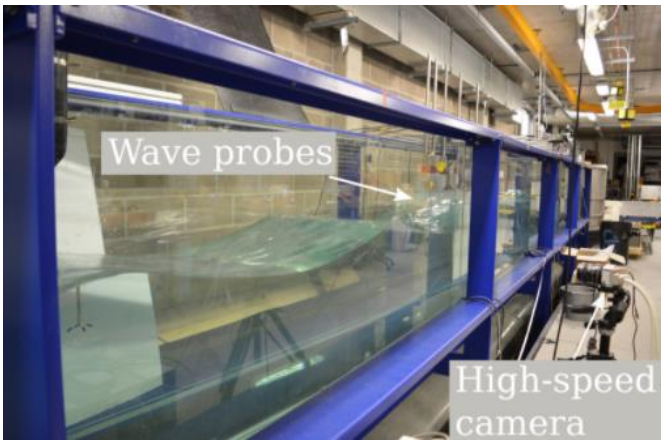


Figure 16 Down wave view [7]

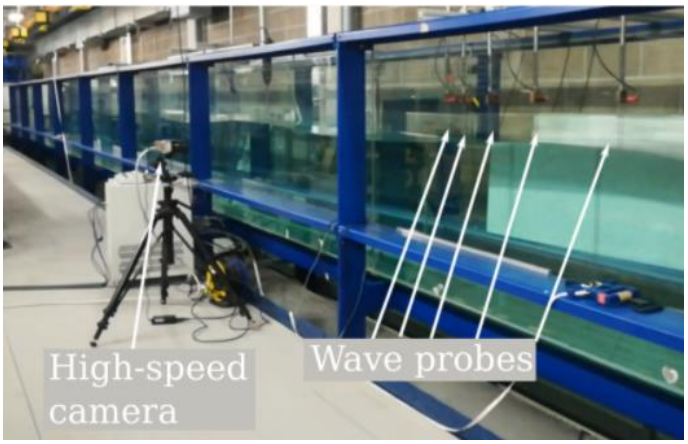


Figure 17 Up wave view [7]

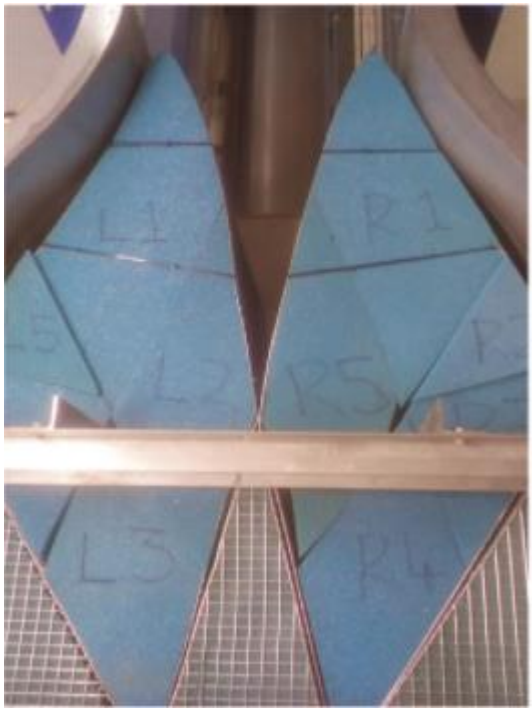


Figure 18 Up view of the beach construction[20]

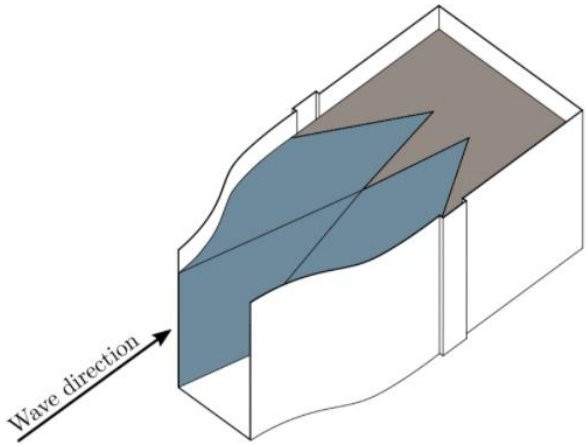


Figure 19 Schematics of the beach [20]

4.2 Numerical wave tank

The use of the numerical wave tank has already been proven in [15], so to model the experiment in OpenFoam, many other parameters must be taken into consideration. Therefore, 3 different areas have been identified: the wavemaker piston, the free surface, and the absorbing beach.

4.2.1 Wavemaker

The wavemaker is a function made by ihFoam, which allows the wall to act like a piston. More information of the contribution of ihFoam can be found in [12]. It has been implemented in the pointDisplacement file, located in the 0 folder.

```
leftwall
{
    type            waveMaker;
    value           uniform (0 0 0);

    motionType      piston;
    x0              (0 0 0);
    n               (1 0 0);
    waveHeight      0.025;
    initialDepth    0.7;
    wavePeriod      1.53;
    rampTime        2.0;
    wavePhase       0;
}
```

Figure 20 Creation of the piston

Where :

- n is the direction, in which the patch will move
- waveHeight is H
- initialDepth is d
- wavePeriod is T

This tool is available since the version 1912 of OpenFoam.

4.2.2 Convergence study

In order to postprocess the results, a good mesh resolution must be implemented. However, a finer mesh results in greater computational resources and thus time, so a convergence study has to be carried out in order to find a balance between finer results and computational costs. Because it is not needed to have the 30 meters tank implemented to proceed to the convergence study, a 5-meter tank has been modelled with different mesh sizes. According to work [15], to the work [7] and the work [17], the free surface elevation and the velocity were the criteria to define a fine mesh with good results.

The following method was used:

- (a) Throughout the course of the simulation, extract the $\eta(x_n,t)$ at every meters of the tank and extract the horizontal velocity
- (b) Determine the time T, where there is a wave crest or a wave trough
- (c) Decrease the time step dT, to get more data to proceed
- (d) Proceed to the following calculations:
    - The calculations need that at least 3 studies were carried out. We will define *f* as the finer mesh, *mf* as medium fine mesh and *c* as coarse mesh
    - S is defining the results, that are compared. Thus S is either the horizontal velocity, either the free surface elevation
    - Calculate

$$\varepsilon_{f-mf} = S_f - S_{mf}$$

$$\varepsilon_{mf-c} = S_{mf} - S_c$$

$$R = \frac{\varepsilon_{f-mf}}{\varepsilon_{mf-c}}$$

- (e) At this point, 3 conditions of convergence are possible:
    - Monochromatic convergence: $0 < R < 1$

- Oscillatory convergence: R < 0
- Divergence: |R| > 1

6 different meshes were created in order to carry out this convergence study. Each finer mesh has a smaller size of cell, divided by 4 (2 in verticals and 2 in horizontal point of view). Finer mesh is used near the free surface. The pictures below show the coarsest and the finest meshes and indicates the cell cize:

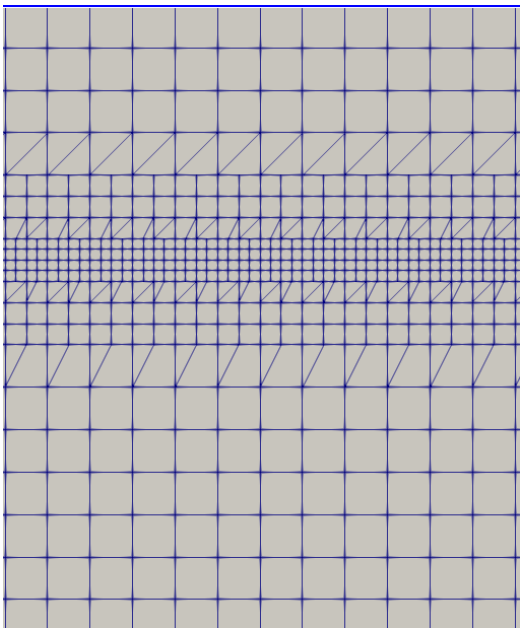


Figure 21 coarsest mesh, minimum cell size : 1.25 x 1.25 cm

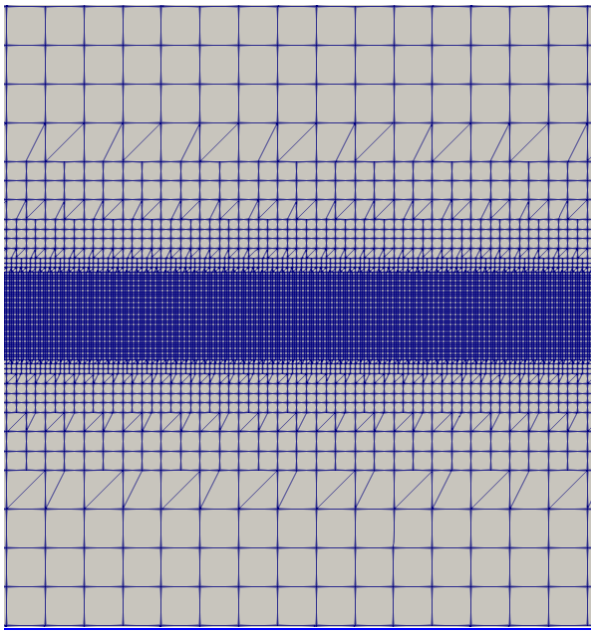


Figure 22 finest mesh, minimum cell size : 7.8125e-2 x 7.8125e-2 cm

Mesh size (in cm)	Level of refinement
5	base mesh
2.5	1 level
1.25	2 level
0.625	3 level
0.3125	4 level
0.15625	5 level
0.078125	6 level

Table 1 Different mesh size, different computational cost

According to the protocol, in order to identify crests and trough, numerical wave probes were used, implemented in the controlDict file (released in the version 2006 of OpenFoam).

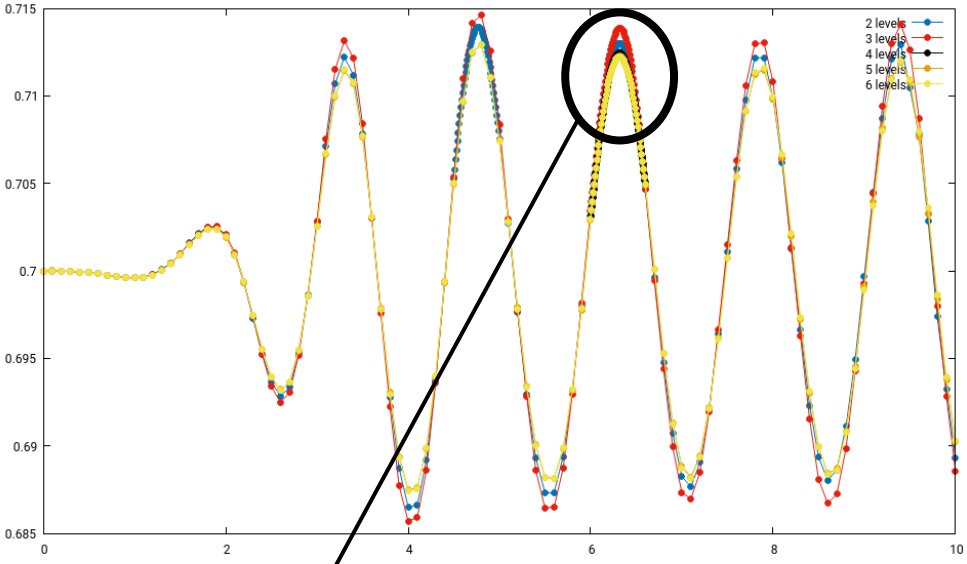


Figure 23 Free surface elevation

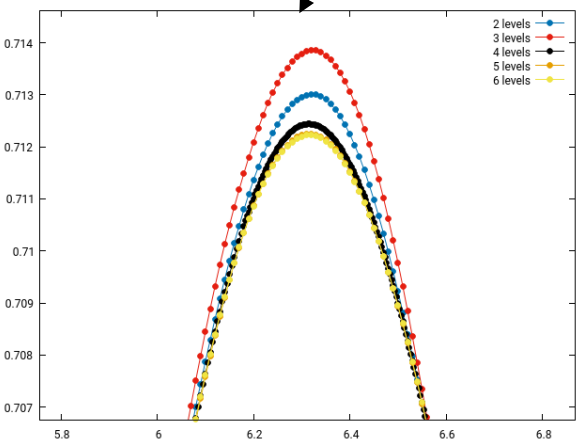


Figure 24 Zoom on a crest

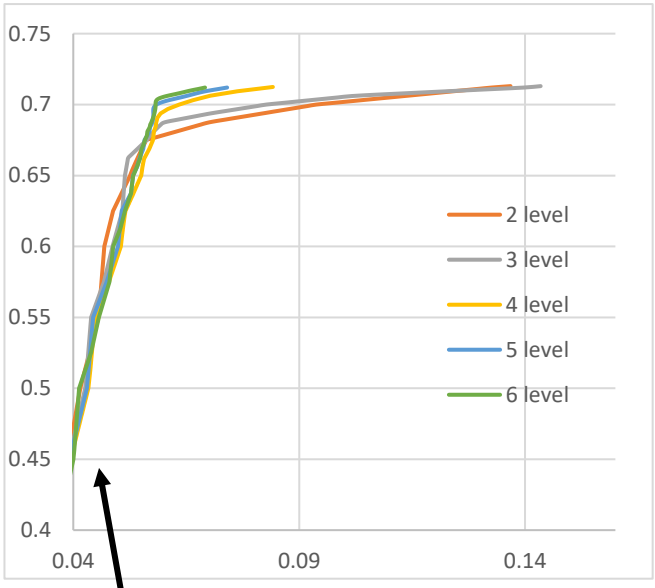


Figure 25 Zoom on the velocity profile

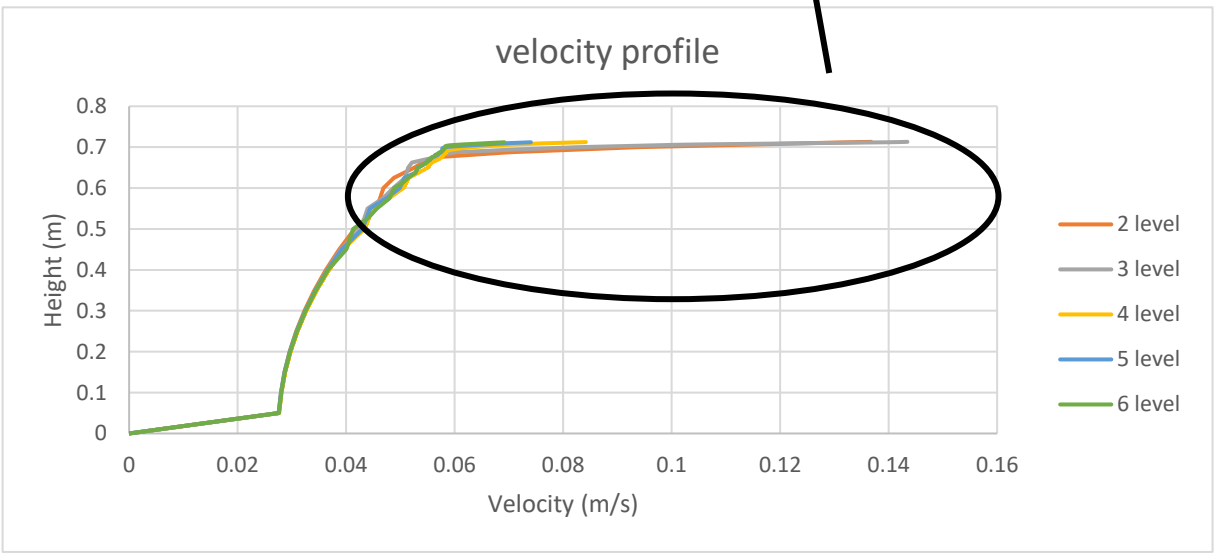


Figure 26 Velocity profile

At this point, we can clearly see that the results are convergent for the. But to select which level to choose, the use of the R coefficient is needed. Therefore the table below summarize the protocol's calculations.

Convergence on alpha.water

Level of refinement	6	5	4	3	2
Free surface elevation	0.7122251	0.7122561	0.7124362	0.7138642	0.7130103
ε	3-2	4-3	5-4	6-5	
	0.0008539	-0.001428	-0.0001801	-3.1E-05	
R	R432	R543	R654		
	-1.67232697	0.12612045	0.1721266		

Convergence on U horizontal

Level of refinement	6	5	4	3	2
Horizontal velocity	0.069166	0.074056	0.084188	0.14346	0.13679
ε	3-2	4-3	5-4	6-5	
	0.00667	-0.059272	-0.010132	-0.00489	
R	R432	R543	R654		
	-8.8863568	0.17094075	0.48262929		

Based on this study and on the protocol of [17], we can conclude that both velocity and FSE are converging, R < 1. The level of mesh refinement adopted is then the 5th.

4.2.3 Modelling the beach

In the real experiment, the beach was constructed in order to absorb the wave. But there were still reflective waves after the beach. A reflection coefficient was then calculated. It is defined as the ration between incident and reflected wave spectra. Those coefficients are depicted in the table below:

Case ID	Reflection coefficient [%]
I1	11.6
I2	12.4
I3	14.5
I4	16.8
I5	18.5
D1	7.3
D2	7.5
D3	8.2
D4	9.1
D5	13.8

Figure 27 Reflection coefficients [20]

In order to create a numerical absorbing and reflecting condition, many theories have been studied and fully developed in the reference [10]

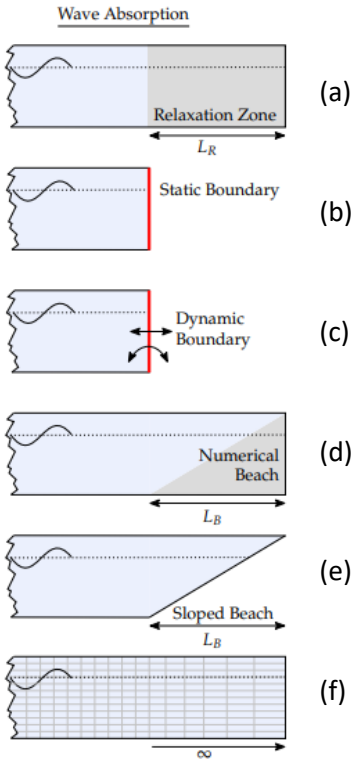


Figure 28 Different methods for modelling the beach [15]

- (a) In the relaxation method zone, a target solution is applied at the boundary of the NWT and a weighting function is implemented in the relaxation, thus providing a smooth transition from the computed solution to the target solution
- (b) In the static boundary method, a correction velocity is applied on the boundary in order to cancel the incident wave field.
- (c) The dynamic boundary method is a method where the boundary of the NWT is moving at a controlled velocity, provided by a force feedback or analytical expression or directly by the physical wave tank measurements.
- (d) The numerical beach method adds an absorbing term in the RANS equation

$$\frac{\partial \rho * U(x,t)}{\partial t} + \nabla \cdot \rho * U(x,t)U(x,t) = -\nabla p(x,t) + \nabla \cdot T(x,t) + \rho f_b(x,t) + S(x)\rho U(x,t) \quad (1)$$

The S(x)ρU(x,t), introduced by this equation, is used to dissipate the wave.

- (e) The sloped bathymetry method does not include a term in the equation, but rather changes the domain layout. It replicates more the physical world.
- (f) The mesh stretching does not require a term in the governing equation. The spatial stretching of the cells here plays the role of absorber, so any wavelength shorter than the cell size are entirely filtered

out. Therefore the spatial domain is big, but the cell count is not dramatically increased due to the expansion of the cell.

For this study, the numerical beach method was used, because, with a fixed beach size, only 1 parameter will change in order to achieve different reflection coefficients.

4.2.4 Corrected sand function

As it is implied in the equation (1), a new term must be added to the Navier-Stokes equation. This term is detailed in the work [13] and the sand coefficient S(x) is given as :

$$S(x) = -2sand_{Max}\left(\frac{l_{beach}-x}{l_{beach}}\right)^3 + 3sand_{Max}\left(\frac{l_{beach}-x}{l_{beach}}\right)^2 \quad (2)$$

Where l_{beach} is the length of the numerical beach, x is the position within the numerical beach, equalling zero at the start and increasing to l_{beach} at the NWT wall, and $sand_{Max}$ stands for the maximum value of sand. According to this work [13], the value of sand should be increased gradually from the start to the end of the numerical beach, in order to avoid a sharp difference in absorption and thus avoid numerical reflection.

However, by applying this equation (2) and plotting it for random values of $sand_{Max}$ and l_{beach}, we can clearly see that instead of gradually increasing the sand coefficient, this function is creating a step function and decreasing gradually as shown in the figure below:

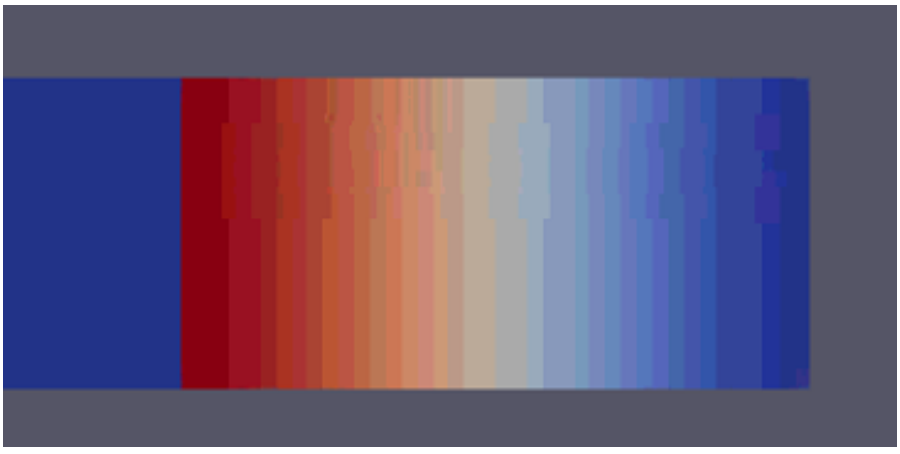


Figure 29 Sand coefficient, with l_{beach} =2 m, $sand_{Max}$ = 15, the blue parts stand for S(x) = 0 and the red to S(x) = 15.

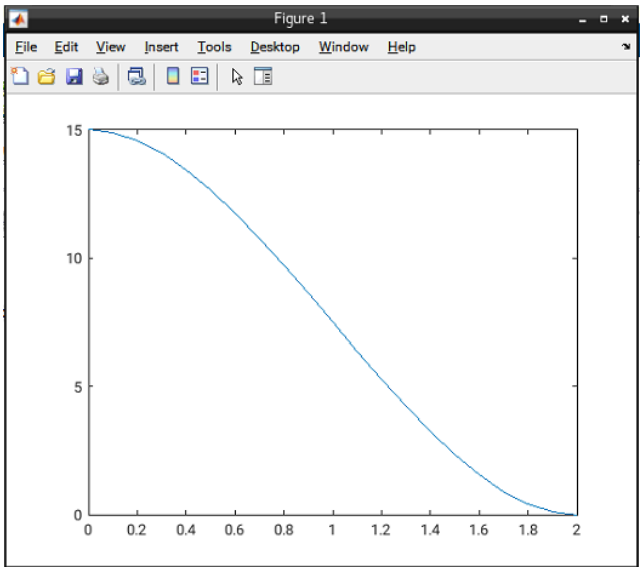


Figure 30 Plotting of the sand function with l_{beach} = 2 m, $sand_{Max}$ = 15

This function will thus create numerical reflection. Therefore, another function must be created to avoid this problem:

$$S(x) = 2sand_{Max}\left(\frac{l_{beach} - x}{l_{beach}}\right)^3 - 3sand_{Max}\left(\frac{l_{beach} - x}{l_{beach}}\right)^2 + sand_{Max}$$

By using this equation, we stick to the description given in [13]: the sand coefficient is gradually increasing from 0 at the beginning of the beach and reaching $sand_{Max}$ as shown in the figures below.

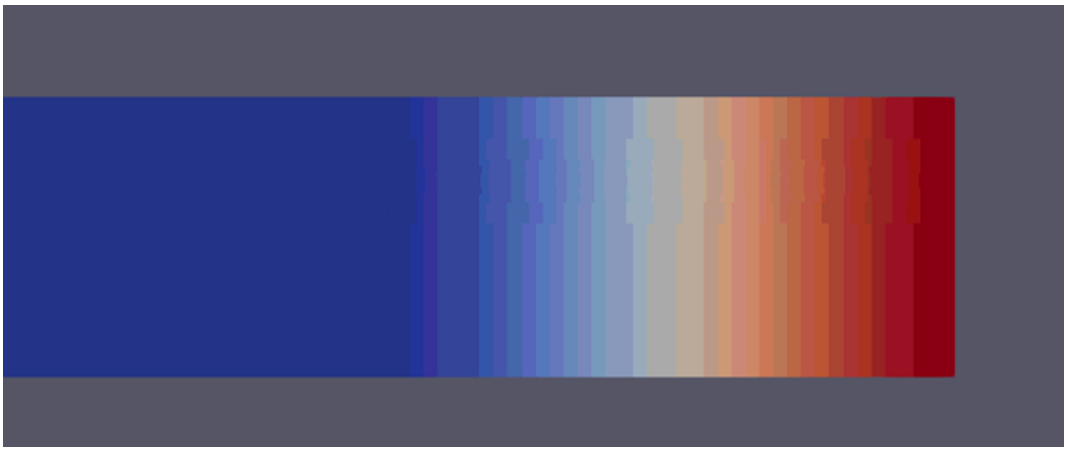


Figure 31 Corrected sand coefficient where the blue part stands for S(x) = 0 and the red part stands for S(x) = $sand_{Max}$

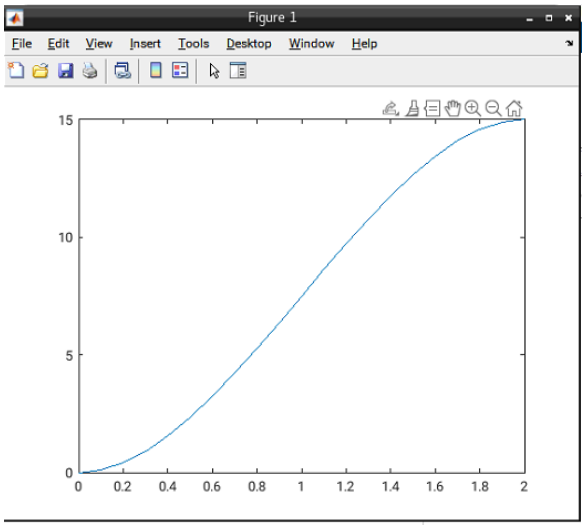


Figure 32 Correted sand coefficient function, with l_{beach} = 2m, $sand_{Max}$ = 15

In order to create this function on OpenFOAM, a tool has to be added: swak4Foam. Indeed, this library offers the user the possibility to specify expressions involving the fields and evaluates them. More information about this utility can be found in the contribution site [9].

The function will then be implemented in the funckySetFieldsDict file, located in the system folder

```
expressions
(
        sand0
        {
                variables
                (
                        "z=pos().z;"
                        "x=pos().x-10;"
                        "sandM=15;"
                        "lbeach=2;"
                );
                field sand;
                expression "2*sandM*pow((lbeach-x)/lbeach,3)-3*sandM*pow((lbeach-x)/lbeach,2)+sandM";
                condition "x>0";
        }
);
```

Figure 33 Content of the funckySetFieldsDict for a NWT of 12 m long

4.3 InterFoamBeach

A new solver has to be created in order to handle the numerical beach. It uses the same parameters as the interFoam solver, which is originally provided in OpenFOAM 2006, with the extra term. The new fields are added in the createFields.H file, located in the main folder of the solver and in the interMixingFoam sub-folder. As said in the work of Christian Windt et al. [10], the

beach is acting on the vertical, z-direction, to dissipate the waves, while allowing for a steady current flow in x-direction.

```
volVectorField zVector
(
    IOobject
    (
    "zVector",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::AUTO_WRITE
    ),
    mesh,
    Foam::vector(0,0,1)
);

volVectorField Beach
(
     IOobject
     (
         "Beach",
         runTime.timeName(),
         mesh,
         IOobject::NO_READ,
         IOobject::AUTO_WRITE

     ),
            sand*zVector*rho
);
```

Figure 34 New fields created for the new solver. Introducing the zVector field and the Beach field

Now that the fields are created, the new term can be added in the equation. It is implemented in the UEqn.H file, located in the main directory:

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

UEqn.relax();
```

Figure 35 Addition of the new term to the existing equation

4.4 Reflection coefficient

The simulation is now following the equation (). To calculate the wave reflection coefficient, the three probe method [11] was implemented. A description of this method is given below.

The wave profile observed at any one of the probe positions may be given as a summation of discrete, harmonically related Fourier components:

$$\eta_p(t) = \sum_{k=1}^{N} A_{p,k} * \sin\left(\frac{2 * \pi * k * t}{T} + \alpha_{p,k}\right)$$

Where :

- $A_{p,k}$ is the Fourier coefficient for frequency k/T,
- T is the length of the wave profile which is being observed; thus, the fundamental is 1/T
- $\alpha_{p,k}$, is the phase
- N is an upper limit of summation which depends on the maximum significant frequency component in the series.

The Fourier coefficients and their phases are obtained from a Fourier transform of the FSE function and are given in polar form as:

$$B_{p,k} = A_{p,k} e^{i\alpha_{p,k}}$$

But the wave profile can also be written a sum of an incident wave and a reflected wave which is described by the following equation of a progressive wave:

$$\eta_p(t) = \sum_{k=1}^{N} C_{I,k} * \sin\left(\frac{-2\pi * k * t}{T} + \frac{2\pi * X1}{L_k} + \theta_k\right) + \sum_{k=1}^{N} C_{R,k} * \sin\left(\frac{-2\pi * k * t}{T} + \frac{2\pi * (X1 + 2 * XR1 - X1P)}{L_k} + \theta_k + \varphi_k\right)$$

Where :

- $C_{I,k}$ is the amplitude of the incident wave
- $C_{R,k}$ is the amplitude of the reflected wave
- X1 is the distance between the left wall and the first wave probe
- XR1 is the distance between the right wall and the first wave probe
- X1P is the distance between the probe 1 and the probe p (X11 =0)
- L_k is the wave length of the wave

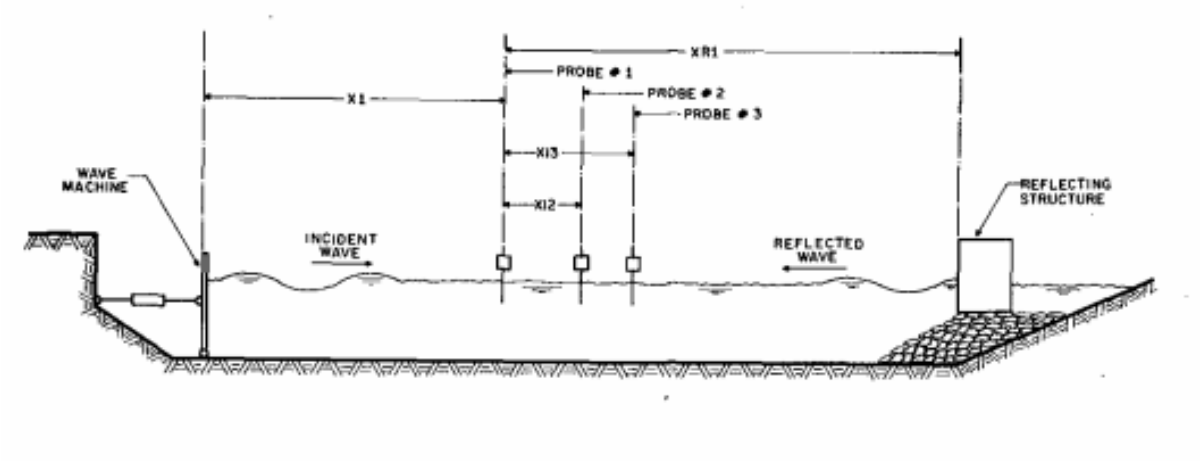


Figure 36 Set up for wave reflection measurement [11]

Let us introduce the following parameters:

$$Z_{I,k} = C_{I,k} * e^{i*(\frac{2\pi*X1}{L_k}+\theta_k)}$$

$$Z_{R,k} = C_{R,k} * e^{i*(\frac{2\pi*(X1+2*XR1)}{L_k}+\theta_k)}$$

Consequently:

$$B_{p,k} = Z_{I,k} * e^{i*\frac{2\pi*X1p}{L_k}} + Z_{R,k} * e^{-i*\frac{2\pi*X1p}{L_k}}$$

According to Mansard and Funke [11] :

$$Z_{I,k} = \frac{1}{D_k} * \Big(B_{1,k} * (R1 + i * Q1) + B_{2,k} * (R2 + i * Q2) + B_{3,k} * (R3 + i * Q3)\Big)$$

$$Z_{R,k} = \frac{1}{D_k} * \Big(B_{1,k} * (R1 - i * Q1) + B_{2,k} * (R2 - i * Q2) + B_{3,k} * (R3 - i * Q3)\Big)$$

Where:

- $\beta_k = \frac{2\pi*X12}{L_k}$
- $\gamma_k = \frac{2\pi*X13}{L_k}$
- $D_k = 2 * (sin^2(\beta_k) + sin^2(\gamma_k) + sin^2(\gamma_k - \beta_k))$
- $R1_k = sin^2(\beta_k) + sin^2(\gamma_k)$
- $Q1_k = \sin(\beta_k)\cos(\beta_k) + \sin(\gamma_k)\cos(\gamma_k)$
- $R2_k = \sin(\gamma_k)\sin(\gamma_k - \beta_k)$
- $Q2_k = \sin(\gamma_k)\cos(\gamma_k - \beta_k) - 2\sin(\beta_k)$
- $R3_k = -\sin(\beta_k)\sin(\gamma_k - \beta_k)$
- $Q3_k = \sin(\beta_k)\cos(\gamma_k - \beta_k) - 2\sin(\gamma_k)$

The reflection coefficient is then evaluated from

$$R(k) = \frac{|Z_{R,k}|}{|Z_{I,k}|}$$

In the appendix A and B, you can find the code that was used to calculate the reflection coefficient. I will explain it in the next section.

```
% limit to 0.05 < kx < 0.45
kmax = 0.45 * 2 * pi / min(d12,d23);
fmax = sqrt( g * kmax * tanh(kmax * h)) / (2.0 * pi);
kmin = 0.05 * 2 * pi / d13;
fmin = sqrt( g * kmin * tanh(kmin * h)) / (2.0 * pi);
```

Figure 37 Study window in term of frequency

This part of the code focuses on the study window of the simulation in term of frequency. It uses the equation of the dispersion for Airy waves, which is a good first approach for this study:

$$f = \frac{\sqrt{g * k * \tanh(k * h)}}{2\pi} \tag{3}$$

```
%apply fourier transform to signals
y1_fft = fft(y1);
y2_fft = fft(y2);
y3_fft = fft(y3);

y1_fft(1) = []; %remove offset
y2_fft(1) = []; %remove offset
y3_fft(1) = []; %remove offset
B(:,1)=y1_fft(mfreq:nfreq)/(0.5*n);
B(:,2)=y2_fft(mfreq:nfreq)/(0.5*n);
B(:,3)=y3_fft(mfreq:nfreq)/(0.5*n);
```

Figure 38 Fourier transform

This section applies the Fourier transform for the function η described in the Mansard and Funke work [11]. It has to be modified in order to get only one side of the Fourier transform.

```
% calculate wave number
for ifreq           = 1:nfreq-mfreq+1
    frequency(ifreq) = 2 * pi * (mfreq + ifreq - 1) * df;
    sig              = frequency(ifreq);
    num              = sig^2/g;
    fun              = @(var)abs(g*var*tanh(h*var)) - sig^2;
    k(ifreq)         = fzero(fun, num);
end

kx(1,:) = zeros(size(k));
kx(2,:) = k .* d12;%beta
kx(3,:) = k .* d13;%gamma
```

Figure 39 Calculations of the wave number, γ_k and β_k

By using the same relation of dispersion (3), the wave numbers are calculated. In the kx matrix, the first line is only 0, the second is every β_k, the third is every γ_k.

```
%calculations
s1 = sum(exp(complex(0,-2*kx)),1);
s2 = sum(exp(complex(0,2*kx)),1);
s3 = sum(B'.*exp(complex(0,-kx)),1);
s4 = sum(B'.*exp(complex(0,kx)),1);
s5 = s1 .* s2 - 9;
%Adjusted xi and xr to be in accordance with wave direction as in documentation
xr = abs((s2.*s3 - 3*s4) ./ s5);
xi = abs((s1.*s4 - 3*s3) ./ s5);
```

Figure 40 Calculations to get Zi and Zr

The calculations to get Z_i and Z_r are detailed below

$$s_1 = \sum_{j=1}^{3} e^{-2ikx(j)} = 1 + e^{-2i\beta} + e^{-2i\gamma}$$

$$s_2 = \sum_{j=1}^{3} e^{2ikx(j)} = 1 + e^{2i\beta} + e^{2i\gamma}$$

$$s_3 = \sum_{j=1}^{3} B(j) * e^{-ikx(j)} = B_1 + B_2 e^{-i\beta} + B_3 e^{-i\gamma}$$

$$s_4 = \sum_{j=1}^{3} B(j) * e^{ikx(j)} = B_1 + B_2 e^{i\beta} + B_3 e^{i\gamma}$$

$$s_5 = \left(1 + e^{-2i\beta} + e^{-2i\gamma}\right) * \left(1 + e^{2i\beta} + e^{2i\gamma}\right) - 9$$

By developing s_5 and writing it in the rectangular form, we have :

$$s_5 = 2 * [cos^2(\beta) - sin^2(\beta) + cos^2(\gamma) - sin^2(\gamma) + cos^2(\gamma - \beta) - sin^2(\gamma - \beta)] - 6$$

$$s_5 = 2 * [3 - 2sin^2(\beta) - 2sin^2(\gamma) - 2sin^2(\gamma - \beta)] - 6$$

$$s_5 = -4 * (sin^2(\beta) + sin^2(\gamma) + sin^2(\gamma - \beta)$$

By using the same notation as Mansard and Funke [11], we have:

$$s_5 = -2 * D_k$$

Next, let us prove that xr is Z_r.

$$x_r(j) = \frac{s_2(j) * s_3(j) - 3 * s_4(j)}{s_5(j)}$$

$$x_r = \frac{\left(1 + e^{2i\beta} + e^{2i\gamma}\right) * \left(B_1 + B_2 e^{-i\beta} + B_3 e^{-i\gamma}\right) - 3 * (B_1 + B_2 e^{i\beta} + B_3 e^{i\gamma})}{-2 * D_k}$$

$$x_r = \frac{B_1 * \overbrace{\left(1 + e^{2i\beta} + e^{2i\gamma} - 3\right)}^{x_1'} + B_2 * \overbrace{\left(e^{-i\beta} + e^{i\beta} + e^{2i\gamma - i\beta} - 3e^{i\beta}\right)}^{x_2'} + B_3 \overbrace{\left(e^{-i\gamma} + e^{i\gamma} + e^{2i\beta - i\gamma} - 3e^{i\gamma}\right)}^{x_3'}}{-2 * D_k}$$

$$x_1' = -2 + \cos(2\beta) + \cos(2\gamma) + i * (\sin(2\beta) + \sin(2\gamma))$$

$$x_1' = -2 * [sin^2(\beta) + sin^2(\gamma) + i * (\sin(\beta)\cos(\beta) + \sin(\gamma)\cos(\gamma)]$$

By using the same notation as Mansard and Funke [11]:

$$x_1' = -2 * (R_1 - i * Q_1)$$

$$x_2' = -\cos(\beta) + \cos(2\gamma - \beta) + i * (-3\sin(\beta) + \sin(2\gamma - \beta))$$

$$x_2' = -\cos(\beta) + \cos(\beta) * \left(1 - 2sin^2(\gamma)\right) + 2\sin(\gamma)\sin(\beta)\cos(\gamma) + i * (-3\sin(\beta) + \sin(2\gamma)\cos(\beta) - \sin(\beta)\cos(2\gamma))$$

$$x_2' = -2\sin(\gamma) * (\sin(\gamma)\cos(\beta) - sin(\beta)\cos(\gamma)) + i * (-4\sin(\beta) + 2\sin(\gamma)[\cos(\gamma)\cos(\beta) - \sin(\beta)\sin(\gamma)])$$

$$x_2' = -2\sin(\gamma) * \sin(\gamma - \beta) + 2i * (-2\sin(\beta) + \sin(\gamma)\cos(\gamma - \beta))$$

By using the same notation as Mansard and Funke [11]:

$$x_2' = -2(R_2 - i * Q_2)$$

Similarly:

$$x_3' = 2\sin(\beta) * \sin(\gamma - \beta) + 2i * (-2\sin(\gamma) + \sin(\beta)\cos(\gamma - \beta))$$

By using the same notation as Mansard and Funke [11]:

$$x_3' = -2(R_3 - i * Q_3)$$

So:

$$x_r = \frac{B_1 * \left(-2 * (R_1 - i * Q_1)\right) + B_2 * (-2(R_2 - i * Q_2)) + B_3 * (-2(R_3 - i * Q_3))}{-2 * D_k}$$

$$x_r = \frac{B_1 * (R_1 - i * Q_1) + B_2 * (R_2 - i * Q_2) + B_3 * (R_3 - i * Q_3)}{D_k}$$

$$x_r = Z_r$$

Let us prove now that x_i is Z_i.

$$x_i(j) = \frac{s_1(j) * s_4(j) - 3 * s_3(j)}{s_5(j)}$$

$$x_i(j) = \frac{\left(1 + e^{-2i\beta} + e^{-2i\gamma}\right) * \left(B_1 + B_2 e^{i\beta} + B_3 e^{i\gamma}\right) - 3 * (B_1 + B_2 e^{-i\beta} + B_3 e^{-i\gamma})}{-2 * D_k}$$

$$x_i = \frac{B_1 * \overbrace{\left(1 + e^{-2i\beta} + e^{-2i\gamma} - 3\right)}^{x_1'} + B_2 * \overbrace{\left(e^{-i\beta} + e^{i\beta} + e^{-2i\gamma+i\beta} - 3e^{-i\beta}\right)}^{x_2'} + B_3 \overbrace{\left(e^{-i\gamma} + e^{i\gamma} + e^{i\beta-2i\gamma} - 3e^{-i\gamma}\right)}^{x_3'}}{-2 * D_k}$$

$$x_1' = -2 + \cos(2\beta) + \cos(2\gamma) - i * (\sin(2\beta) + \sin(2\gamma))$$

$$x_1' = -2 * [sin^2(\beta) + sin^2(\gamma) - i * (\sin(\beta)\cos(\beta) + \sin(\gamma)\cos(\gamma)]$$

By using the same notation as Mansard and Funke [11]:

$$x_1' = -2 * (R_1 + i * Q_1)$$

$$x_2' = -\cos(\beta) + \cos(2\gamma - \beta) - i * (-3\sin(\beta) + \sin(2\gamma - \beta))$$

$$x_2' = -\cos(\beta) + \cos(\beta) * \left(1 - 2sin^2(\gamma)\right) + 2\sin(\gamma)\sin(\beta)\cos(\gamma) - i * (-3\sin(\beta) + \sin(2\gamma)\cos(\beta) - \sin(\beta)\cos(2\gamma))$$

$$x_2' = -2\sin(\gamma) * (\sin(\gamma)\cos(\beta) - sin(\beta)\cos(\gamma)) - i * (-4\sin(\beta) + 2\sin(\gamma)\,[\cos(\gamma)\cos(\beta) - \sin(\beta)\sin(\gamma)])$$

$$x_2' = -2\sin(\gamma) * \sin(\gamma - \beta) - 2i * (-2\sin(\beta) + \sin(\gamma)\cos(\gamma - \beta))$$

By using the same notation as Mansard and Funke [11]:

$$x_2' = -2(R_2 - i * Q_2)$$

Similarly:

$$x_3' = 2\sin(\beta) * \sin(\gamma - \beta) - 2i * (-2\sin(\gamma) + \sin(\beta)\cos(\gamma - \beta))$$

$$x_3' = -2(R_3 - i * Q_3)$$

So

$$x_i = \frac{B_1 * \left(-2 * (R_1 + i * Q_1)\right) + B_2 * (-2(R_2 + i * Q_2)) + B_3 * (-2(R_3 + i * Q_3))}{-2 * D_k}$$

$$x_i = \frac{B_1 * (R_1 + i * Q_1) + B_2 * (R_2 + i * Q_2) + B_3 * (R_3 + i * Q_3)}{D_k}$$

$$x_i = Z_i$$

So with this algorithm, we can calculate the reflection coefficient of the sand and therefore get the same coefficient as in the real experiment.

4.5 Results

The further studies are focused on the case I1 of the real experiment.

Many simulations were run in order to find the correct value of the sand coefficient. Here are the results of them

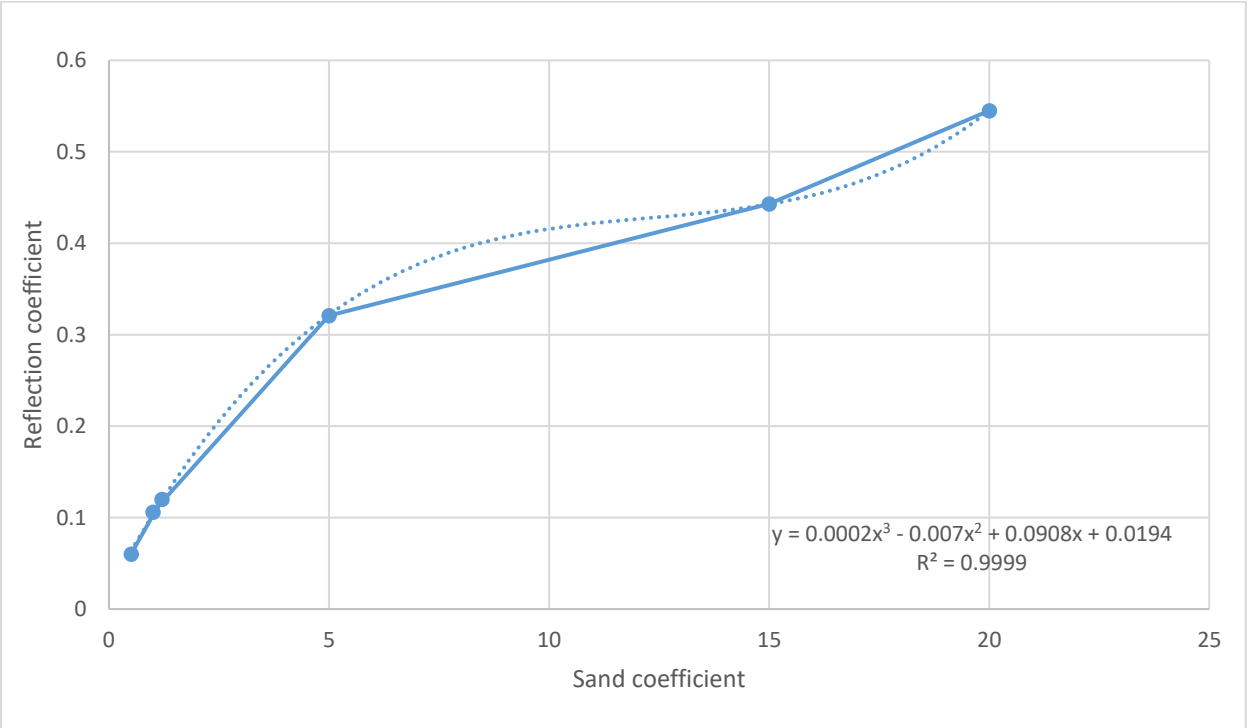


Figure 41Results of the reflection coefficient in function of the sand coefficient

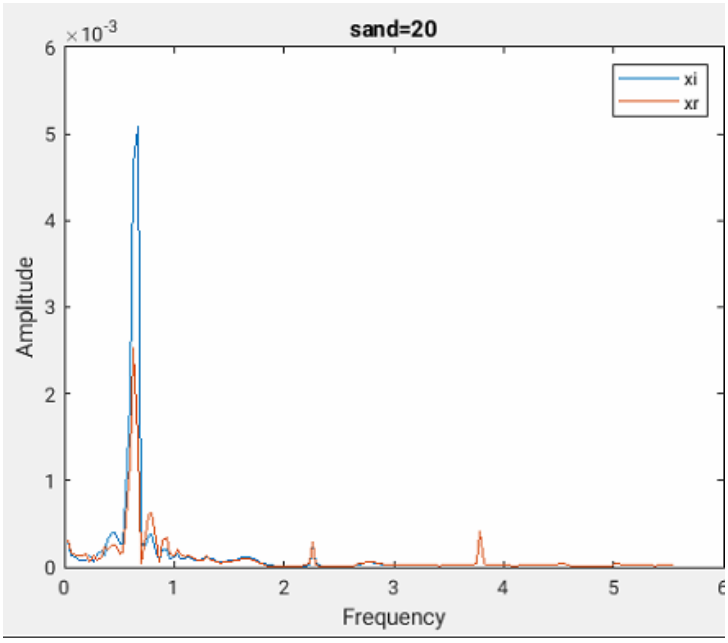
Sand coefficient	Reflection coefficient
20	0.545
15	0.443
5	0.321
1.2	0.118
1	0.1059
0.5	0.06

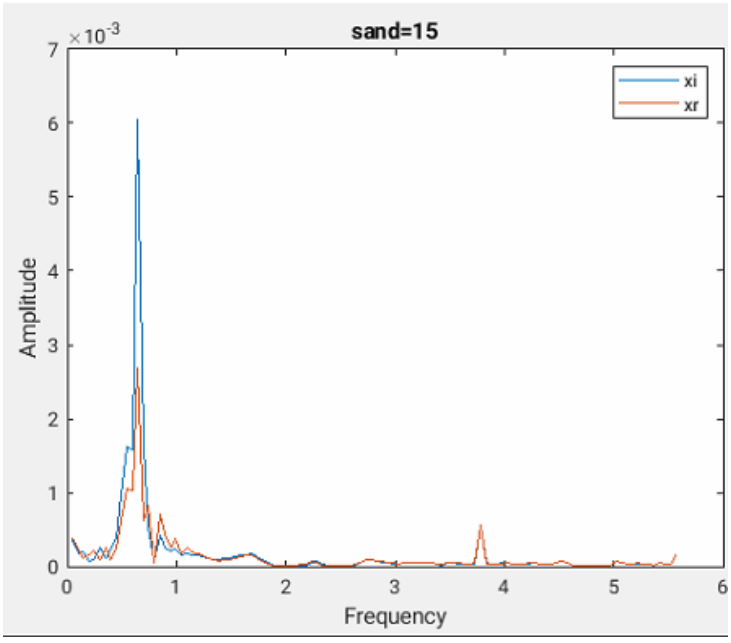
A tendency function has been drawn in the upper graphic:

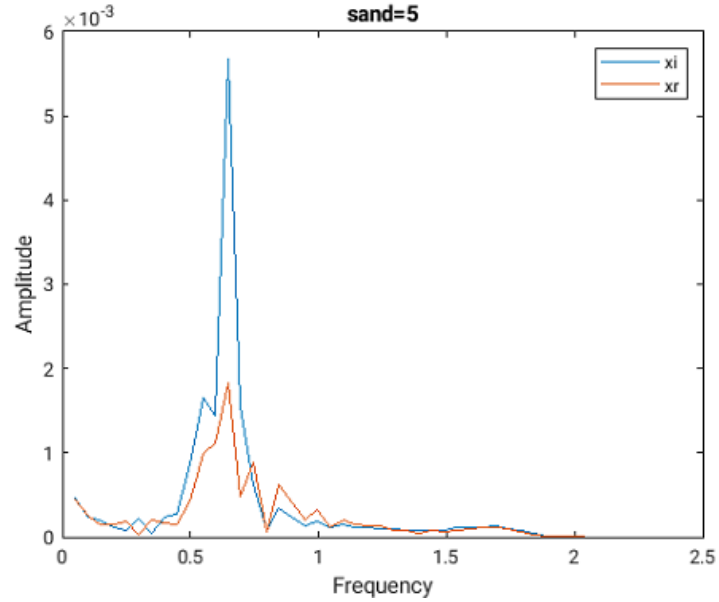
$$reflection\ coefficient = y = f(sand\ coefficient) = f(x)$$

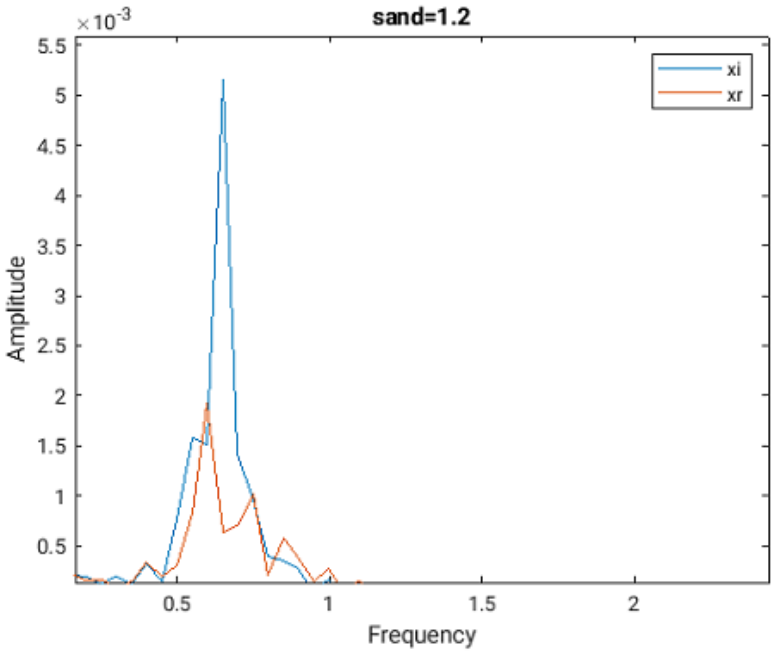
$$y = 2 * 10^{-4} * x^3 - 0.007 * x^2 + 0.0909 * x + 0.0188$$

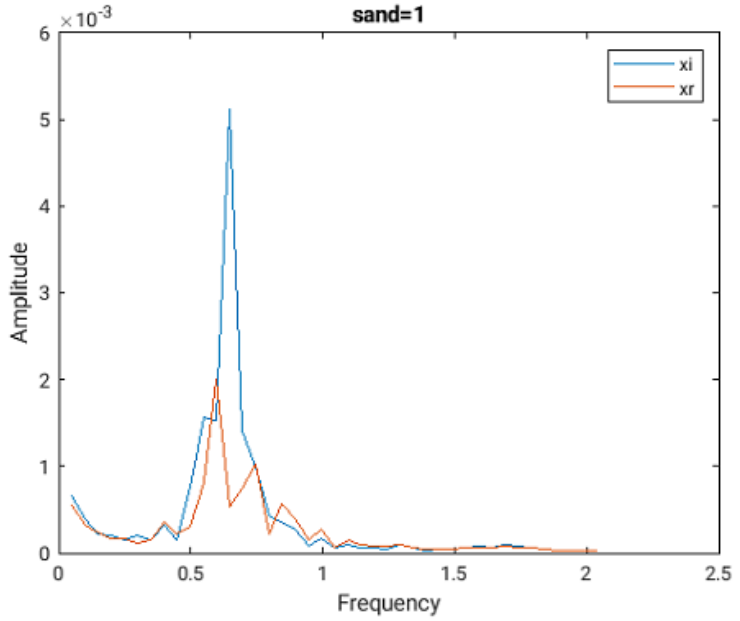
However this formula is not to be considered as a general rule, because not enough simulation were made to confirm this function.











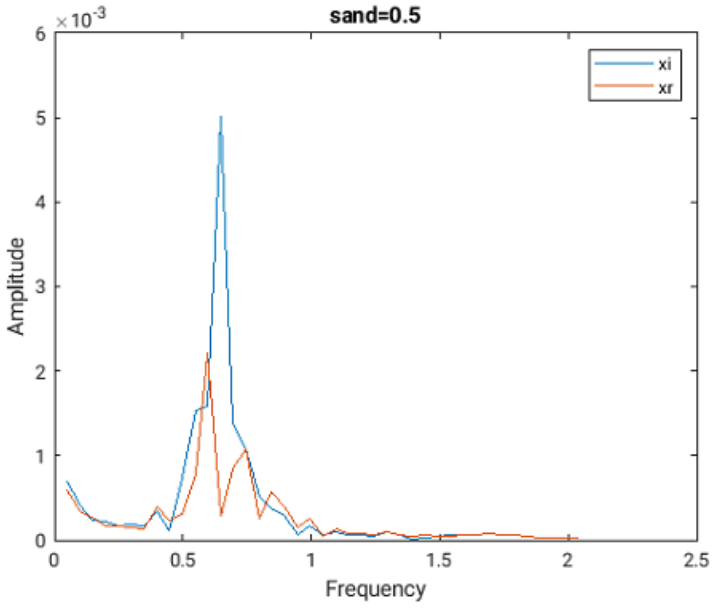


Figure 42 Graphs of Xi and Xr in function of the sand

According to those results, the coherent sand coefficient for the I1 case is around 1.2.

4.6 Analysis of the results

In this section, we will compare the results of the I1 case, the results with the Wheeler method and the extrapolation method.

As for the convergence study, the creist was identified from the datas of the simulation and the results are the following:

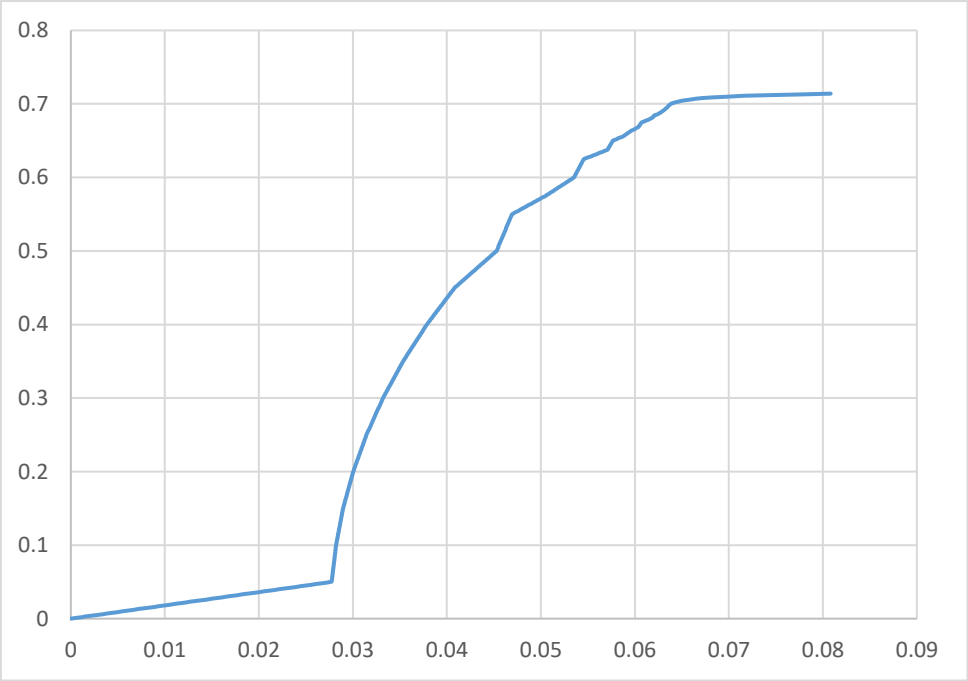


Figure 43 Horizontal velocity

By using the formulas given in previous sections, the Wheeler stretching, and the extrapolation method give the results below

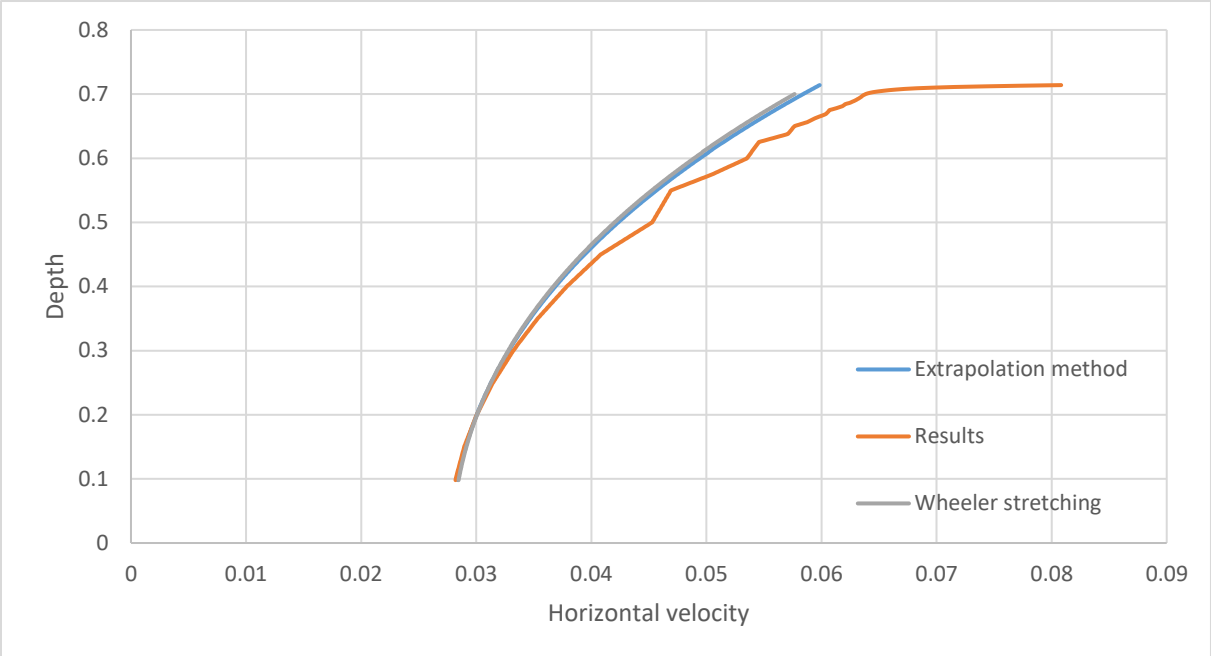


Figure 44 Comparison of the results from different methods and the simulation

The results from the extrapolation method and the Wheeler stretching method differ slightly, but both results differ from the results of the simulation as it is summarized in the table below:

Error wheeler/extrapolation in %	**Error extrapolation/results in %**	**Error wheeler/results in %**
2.38	25.15	26.94

We can clearly see that there is a huge difference between he simulation and the theories. Many explanations can be given in order to explain this difference. The main reason is that the simulation is using the laminar theory which implies that the simulation is running in a steady state. It results in a reasonable computational cost, but with errors due to this decision. An other reason is that both of these theories are not well-suited for intermediate water waves.

5 Conclusion and further work

After modelling the NWT, an optimal mesh refinement was found and a modification was made on the sand coefficient in order to be more coherent with the explanation given by [13]. The sand coefficient was thus found according to the value of the experiment. However the fact that the laminar theory was used resulted in different results compared to the Wheeler stretching method and the extrapolation method.

It is thus necessary to keep the parameters that were found in this study,but to use an other theory for the simulation, as the Reynolds-averaged Navier–Stokes equations (RANS). Moreover, all cases from the experiment were not simulated, so a further study could implement the same parameters as in this study, but for the others cases.

6 References

- [1] H. Versteeg and W. Malalasekera. An introduction to Computational Fluid Dynamics: The Finite Volume Method (2nd edition), 2007.
- [2]Nichols, B.D. and Hirt, C.W., "Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies," Proc. First Intern. Conf. Num. Ship Hydrodynamics, Gaithersburg, ML, Oct. 20-23, 1975
- [3]Hirt, C.W. and Nichols, B.D., "Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries," Journal of Computational Physics 39, 201, 1981.
- [4] Robert Eymard, Thierry Gallouët, Raphaèle Herbin. Finite Volume Methods. J. L. Lions; Philippe Ciarlet. Solution of Equation in ffn (Part 3), Techniques of Scientific Computing (Part 3), 7, Elsevier, pp.713-1020, 2000, Handbook of Numerical Analysis, 9780444503503. ff10.1016/S1570-8659(00)07005- 8ff. ffhal-02100732v2
- [5] OpenFOAM: User Guide: Time schemes https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-time.html
- [6] OpenFOAM: User Guide: Interpolation schemes
- [7] ALIX UNTRAU; Kinematics of regular and focused waves: An experimental and numerical study
- [8] 1 Introduction (openfoam.com)
- [9]Contrib/swak4Foam - OpenFOAMWiki
- [10] Development of an impulse-source-based wave-current interaction model
- [11] Mansard and Funke; The measurement of incident and reflected spectra using a least square method
- [12] Contrib/IHFOAM - OpenFOAMWiki
- [13] Windt, Davidson, Schmitt and Ringwood - 2018 (ECFD) - Development of an impulse source based wave current interaction model
- [14] Robert Eymard, Thierry Gallouët, Raphaèle Herbin. Finite Volume Methods. J. L. Lions; Philippe Ciarlet. Solution of Equation in ffn (Part 3), Techniques of Scientific Computing (Part 3), 7, Elsevier, pp.713-1020,

2000, Handbook of Numerical Analysis, 9780444503503. ff10.1016/S1570-8659(00)07005- 8ff. ffhal-02100732v2f

- [15] Christian Windt , Josh Davidson , Pál Schmitt and John V. Ringwood; On the Assessment of Numerical Wave Makers in CFD Simulations
- [16] N. Veritas, Environmental conditions and environmental loads, Det Norske Veritas (2000)
- [17] Fred Stern, Robert V. Wilson, Hugh W. Coleman*, and Eric G. Paterson; VERIFICATION AND VALIDATION OF CFD SIMULATIONS
- [18] Folley M. (2017) The Wave Energy Resource. In: Pecher A., Kofoed J. (eds) Handbook of Ocean Wave Energy. Ocean Engineering & Oceanography, vol 7. Springer, Cham. https://doi.org/10.1007/978-3-319-39889-1_3
- [19] B. Le Méhauté, an introduction to hydrodynamics and water waves, Springer Science & Business Media, 2013
- [20] Christian Windt, Alix Untrau, Josh Davidson, Edward J. Ransley,Deborah Greavesd, John V. Ringwooda; Assessing the validity of regular wave theory in a short physical wave flume using particle image velocimetry
- [21] Josh Davidson, Marie Cathelain, Louise Guillemet, Thibault Le Huec & John V. Ringwood (2015): Implementation of an OpenFOAM Numerical Wave Tank for Wave Energy Experiments Proceedings of the 11th European Wave and Tidal Energy Conference 6-11th Sept2015, Nantes, France pp.(09B1-1-1)-(09B1-1-10)
- [22] Chapter16.pdf (mit.edu)

7 Appendix A : general code

7.1.1.1 Extracting the results of the simulation

```
Y=readtable('position.dat');
```

7.1.1.2 Constructing the results for the wave probes

```
y1=table2array(Y(:,4));
y1=y1-y1(1);
y2=table2array(Y(:,8));
y2=y2-y2(1);
y3=table2array(Y(:,12));
y3=y3-y3(1);
```

7.1.1.3 Parameters

```
d12=0.3;
d23=0.1;
h=0.025;
dt=0.1;
```

7.1.1.4 Function

```
[frequency, xi, xr] = mon3probe(y1,y2,y3,d12,d23,h,dt);
```

```
Elapsed time is 0.014493 seconds.
```

7.1.1.5 Results

```
r(1,:)=frequency/(2*pi)
r(2,:)=xr./xi;
```

Published with MATLAB® R2020b

8 Appendix B : 3 probes method

```
function [frequency,xi, xr] = mon3probe(y1, y2, y3, d12,d23, h, dt);
```

8.1.1.1 Parameters

```
% constants
g = 9.81;

%number of data points
n = length(y1);

%calculate nyquist frequency
df = 1 / (n*dt);

% separation ratio
d13 = d12+d23;
mu = d12/d13;
```

8.1.1.2 Limit to 0.05 < kx < 0.45

```
kmax = 0.45 * 2 * pi / min(d12,d23);
fmax = sqrt( g * kmax * tanh(kmax * h)) / (2.0 * pi);
kmin = 0.05 * 2 * pi / d13;
fmin = sqrt( g * kmin * tanh(kmin * h)) / (2.0 * pi);

if (fmin<=0)
        error ('(fmin<0)');
    end

nfreq = fix(fmax/df);
mfreq = fix(fmin/df);
if (nfreq<mfreq)
        error ('(nfreq<mfreq)');
    end
if (mfreq<=0)
        error ('(mfreq<0)');
    end
```

8.1.1.3 Apply fourier transform to signals

```
y1_fft = fft(y1);
y2_fft = fft(y2);
y3_fft = fft(y3);

y1_fft(1) = []; %remove offset
y2_fft(1) = []; %remove offset
y3_fft(1) = []; %remove offset
B(:,1)=y1_fft(mfreq:nfreq)/(0.5*n);
B(:,2)=y2_fft(mfreq:nfreq)/(0.5*n);
```

```
B(:,3)=y3_fft(mfreq:nfreq)/(0.5*n);
```

8.1.1.4 Calculate wave number

```
for ifreq                          = 1:nfreq-mfreq+1
   frequency(ifreq) = 2 * pi * (mfreq + ifreq - 1) * df;
   sig                             = frequency(ifreq);
   num                             = sig^2/g;
   fun                             = @(var)abs(g*var*tanh(h*var)) - sig^2;
   k(ifreq)               = fzero(fun, num);
end

kx(1,:) = zeros(size(k));
kx(2,:) = k .* d12;%beta
kx(3,:) = k .* d13;%gamma
```

8.1.1.5 Calculations

```
s1 = sum(exp(complex(0,-2*kx)),1);
s2 = sum(exp(complex(0,2*kx)),1);
s3 = sum(B'.*exp(complex(0,-kx)),1);
s4 = sum(B'.*exp(complex(0,kx)),1);
s5 = s1 .* s2 - 9;
```

8.1.1.6 Adjusted xi and xr to be in accordance with wave direction as in documentation

```
xr = abs((s2.*s3 - 3*s4) ./ s5);
xi = abs((s1.*s4 - 3*s3) ./ s5);
end
```