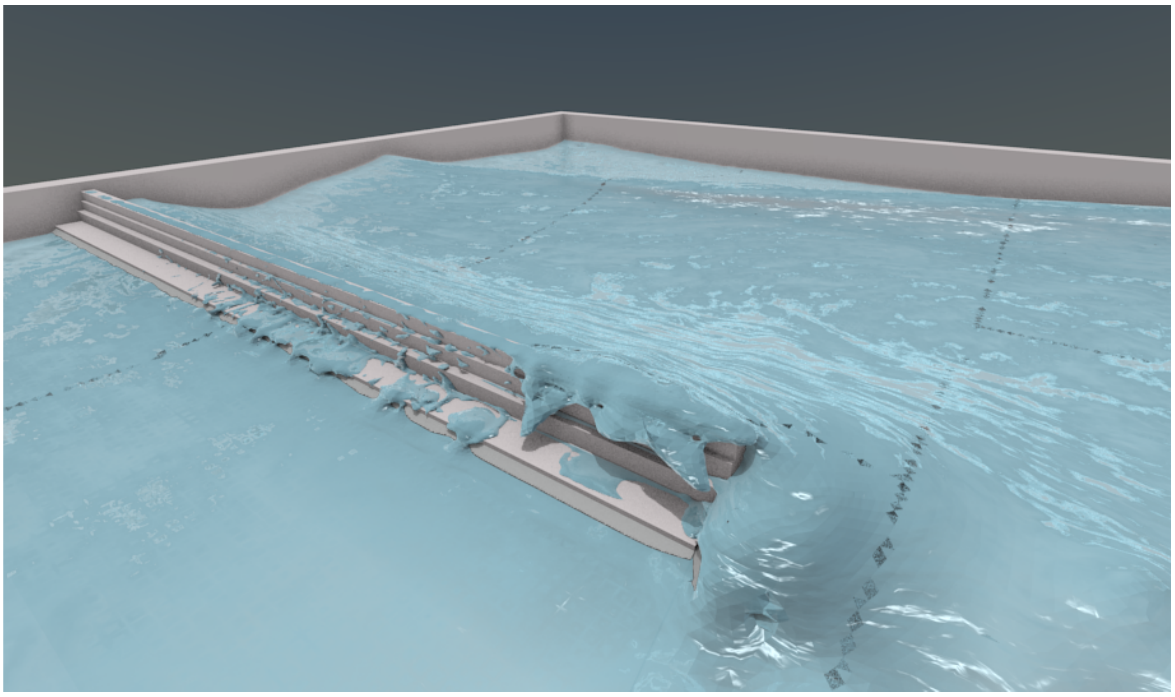


IHFOAM Manual
15th July 2014

Contents

Chapter 1

Introduction to IHFOAM

1.1 Overview

IHFOAM is a two-phase solver based in **interFoam**. It solves the Volume-Averaged Reynolds-Averaged Navier-Stokes (VARANS) equations and it includes built-in boundary conditions for wave generation and active wave absorption.

1.2 ihFoam Solver

1.2.1 Description

ihFoam is one of the solvers included in OpenFOAM®. It solves the three-dimensional Reynolds Averaged Navier-Stokes (RANS) equations for two incompressible phases using a finite volume discretization and the volume of fluid (VOF) method. In VOF (Berberovic et al., 2009), each phase is described by a fraction α_i occupied by the volume of fluid of the $\mathrm{i^{th}}$ material in the cell. Its principal advantages are that it is very simple, allowing very complex free surface configurations to be represented easily and that it involves no mesh motion. A minor disadvantage is that it becomes less effective as surface tension effects increase. However, most of the times in coastal engineering practical applications, we are dealing with relatively long wavelengths, so that only for very specific phenomena are surface tension forces not negligible. It supports several turbulence models (e.g. $k-\epsilon$, $k-\omega$ SST, LES).

ihFoam solver is prepared for static meshes only. An enhanced version of it is **ihDyM-**

Foam, which handles dynamic meshes ("DyM" stands for Dynamic Mesh). Hence it can simulate floating body movements or dynamic mesh refinement along the free surface. Other than that it solves the same equations, following the same procedures.

The boundary condition modules have been implemented to work indistinctly with **ihFoam** or **ihDyMFoam**. These add the capabilities of generating water waves according to multiple wave theories. Additionally, active wave absorption has been programmed based on extrapolated 2D theory, to allow outgoing waves to flow away with minor reflection, either in pure absorbent boundaries or in wave generation boundaries. Furthermore, other theories have been developed in 3D to account for non orthogonal incident waves.

1.2.2 Governing Equations

The aforementioned RANS equations, which include continuity (1.1) and momentum conservation (1.2) equations, are the governing mathematical expressions which link pressure and velocity. The assumption of incompressible fluids has been used, which is applicable for most coastal engineering practical problems.

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0 \tag{1.1}$$

$$\frac{\partial \rho \langle u_i \rangle}{\partial t} + \frac{\partial}{\partial x_j}\left[\frac{1}{\phi}\rho \langle u_i \rangle \langle u_j \rangle\right] = - \phi \frac{\partial \langle p \rangle^f}{\partial x_i} + \phi \rho g_i + \frac{\partial}{\partial x_j}\left[\mu_{eff} \frac{\partial \langle u_i \rangle}{\partial x_j}\right] - [CT] \tag{1.2}$$

where all the bold letters indicate a vector field. All the variables are referenced in table 1.1. The closure terms ($[CT]$) are as follows:

$$[CT] = A \langle u_i \rangle + B \left|\langle u \rangle\right| \langle u_i \rangle + C \frac{\partial \langle u_i \rangle}{\partial t} \tag{1.3}$$

In this work the friction coefficients are calculated according to Engelund (1953) for-

ρ	Density, which is calculated as presented in equation 1.5
$\mathbf{U}, u_i$	Velocity vector
p^*	Pseudo-dynamic pressure
$\mathbf{g}, g_i$	Acceleration due to gravity
$\mathbf{X}$	Position vector
ϕ	Porosity
$\sigma\kappa\nabla\alpha$	Surface tension term
σ	Surface tension coefficient
κ	Curvature of the interface: $\kappa = \nabla \cdot \frac{\nabla\alpha}{\lvert\nabla\alpha\rvert}$
α	Indicator (VOF) function
μ_{eff}	Efficient dynamic viscosity, which takes into account the molecular dynamic viscosity plus the turbulent effects: $\mu_{\text{eff}} = \mu + \rho\,\nu_{\text{turb}}$
ν_{turb}	Turbulent kinetic viscosity, given by the chosen turbulence model

Table 1.1: Variables used in this chapter.

mulas, as applied in Burcharth and Andersen (1995), therefore:

$$A = \alpha \frac{(1-\phi)^3}{\phi^2} \frac{\mu}{D_{50}^2} \tag{1.4a}$$

$$B = \beta \left(1 + \frac{7.5}{\text{KC}}\right) \frac{1-\phi}{\phi^2} \frac{\rho}{D_{50}} \tag{1.4b}$$

The elements in equation 1.2 have a particular disposition: those placed on the left hand side of the equal sign are used in OpenFOAM®to assemble the coefficient matrix, and the ones on the right side are calculated explicitly, and form the independent term of the equations.

An additional equation must also be taken into account to describe the movement of the phases. Since for the vast majority of coastal engineering applications only water and air are present, the following analysis is carried out for those two phases only. For a more general approach the reader is referred to Kissling et al. (2010). As a result of this assumption, just one indicator phase function (α) is needed, and is defined as the quantity of water per unit of volume in each cell. This means that if $\alpha = 1$ the cell is full of water, if $\alpha = 0$ the cell is full of air, and in any other case it belongs to the interface. It is straightforward to calculate any of the properties of the fluid in each cell, just by weighting them by the VOF function. For example, density of the cell is computed as follows:

$$\rho = \alpha\,\rho_{\text{water}} + (1-\alpha)\,\rho_{\text{air}} \tag{1.5}$$

The starting point for the equation which tracks the fluid movement is a classic advection equation:

$$\frac{\partial \alpha}{\partial t} + \frac{1}{\phi}\frac{\partial \langle u_i \rangle \alpha}{\partial x_i} = 0 \tag{1.6}$$

However, some restrictions apply in order to obtain physical results: a sharp interface must be maintained and α must be conservative and bounded between 0 and 1. OpenFOAM®makes use of an artificial compression term $(\nabla \cdot \mathbf{U}_c\,\alpha(1-\alpha))$ instead of using a compressing differencing scheme. This approach is conservative and takes non-zero values only at the interface. Furthermore the flow is not compressed if $\mathbf{U_c}$ is normal to the interface $\left(\frac{\nabla\alpha}{|\nabla\alpha|}\right)$, which points towards greater values of α, and therefore from the air to the water phase. This yields the final expression:

$$\frac{\partial \alpha}{\partial t} + \frac{1}{\phi}\frac{\partial \langle u_i \rangle \alpha}{\partial x_i} + \frac{1}{\phi}\frac{\partial \langle u_{c_i} \rangle \alpha(1-\alpha)}{\partial x_i} = 0 \tag{1.7}$$

in which $|\mathbf{U}_c| = min\,[c_\alpha|\mathbf{U}|, max(|\mathbf{U}|)]$, where the user can specify factor c_α. By default it takes value 1, but it can be greater to enhance the compression of the interface.

The boundedness of this equation is achieved by means of a specially designed solver called MULES (Multidimensional Universal Limiter for Explicit Solution). It makes use of a limiter factor on the fluxes of the discretized divergence term to ensure a final value between 0 and 1. For further reference regarding the governing equations see Rusche (2002).

1.2.3 Solving Procedure

Originally, the solving algorithm was PISO (Pressure Implicit with Splitting of Operators); for a detailed description of the numerical solution of the equations see Kissling et al. (2010). More recent versions of the code have improvements. The new algorithm is called PIMPLE, as it is actually a mixture between PISO and SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithms. Its main structure is inherited from the original PISO, but it allows equation under-relaxation to ensure the convergence of all the equations at each time step. Both algorithms are thoroughly explained and applied for VOF in Jasak (1996).

A detailed flow chart, figure 1.1, has been developed in order to show the full loop for solving each time step. The main loop is presented with darker background. The alpha subcycle and the PIMPLE loop are further developed outside it. Both of them include a basic description of the newly developed boundary conditions. Within it, some variables (e.g. "nAlphaSubCycles", "nCorrectors"...) are defined. These are specified in the program control files and govern the performance of the model's solving procedures.

1.3 Installation and Compilation

The way to install **ihFoam** and **ihDyMFoam** in your computer/cluster is straightforward; just clone the IHFOAM distribution to your local machine.

```
>_ git clone git://github.com/phicau/IHFOAM.git
```

Code updates can be downloaded in the future as follows:

```
>_ git checkout
>_ git pull
```

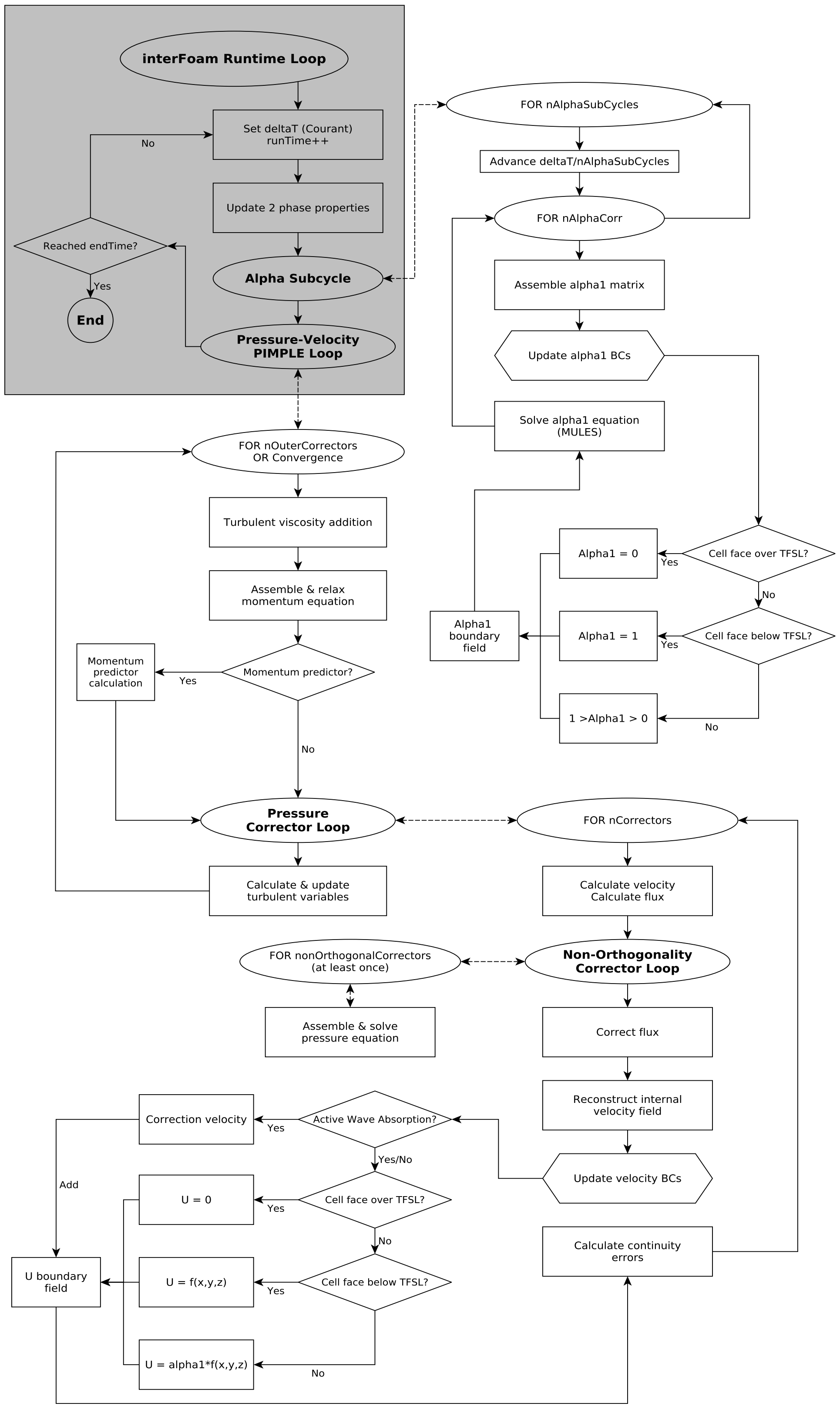
First you need to compile the boundary conditions, run the **allMake** script:

```
>_ cd genAbs
>_ ./allMake
```

And then you can compile the solvers in an identical manner:

```
>_ cd solvers/ihFoamXXXXX
>_ ./allMake
```

Where XXXXX denotes your flavour (OpenFOAM - OF, FOAM-extend - FE) and your version.



interFoam Runtime Loop
Set deltaT (Courant) runTime++
No
Update 2 phase properties
Reached endTime?
Yes
End
Alpha Subcycle
Pressure-Velocity PIMPLE Loop
FOR nAlphaSubCycles
Advance deltaT/nAlphaSubCycles
FOR nAlphaCorr
Assemble alpha1 matrix
Update alpha1 BCs
Solve alpha1 equation (MULES)
Alpha1 = 0
Yes
Cell face over TFSL?
No
Alpha1 boundary field
Alpha1 = 1
Yes
Cell face below TFSL?
1 >Alpha1 > 0
No
FOR nOuterCorrectors OR Convergence
Turbulent viscosity addition
Assemble & relax momentum equation
Momentum predictor calculation
Yes
Momentum predictor?
No
Pressure Corrector Loop
FOR nCorrectors
Calculate & update turbulent variables
Calculate velocity Calculate flux
FOR nonOrthogonalCorrectors (at least once)
Non-Orthogonality Corrector Loop
Assemble & solve pressure equation
Correct flux
Reconstruct internal velocity field
Correction velocity
Yes
Active Wave Absorption?
Yes/No
Add
Update velocity BCs
U = 0
Yes
Cell face over TFSL?
No
Calculate continuity errors
U boundary field
U = f(x,y,z)
Yes
Cell face below TFSL?
U = alpha1*f(x,y,z)
No

Figure 1.1: “interFoam” solving flow chart. “TFSL” stands for Theoretical Free Surface Level.

The boundary conditions can be linked dynamically inside **controlDict** file, so that they work with other solvers (e.g. interFoam). The code needed is as presented next:

```
libs
(
    "libIHwaveGeneration.so"
    "libIHwaveAbsorption.so"
);
```

Chapter 2

Wave Generation Boundary Conditions

In this chapter all the processes related to wave generation are discussed. First, as a global overview, an introductory section is presented. Next, all the wave theories which have been implemented for OpenFOAM®are explained in detail. This includes the theory behind them, and also the specific numerical issues related to the implementation for the model. Finally a more detailed explanation of the code behind the boundary conditions is given. A thorough validation of the boundary conditions is presented in Higuera et al. (2013a) and Higuera et al. (2013b).

2.1 Introduction

Wave generation is a critical element of numerical coastal engineering simulations. Generating waves is usually the beginning of the vast majority of the cases we are dealing with. An accurate wave generation process lays the foundations for realistic final results. If the starting point is not accurate, all the errors introduced in this initial step will propagate until the end. Should this occur, the results may be completely different, as wave interaction can exhibit very important second order effects. Additionally the processes which affect waves, mainly shoaling and breaking, are highly influenced by wave shape and height. If these are not accurately replicated, chances are that the simulation will end up nowhere close to the reality.

Several boundary conditions for wave generation are currently available in literature.

The first and oldest approach is GroovyBC, which is freely available online [1] and distributed independently from OpenFOAM®. While it is not a specific boundary condition for wave generation, it accepts mathematical expressions in an elementary way. As a consequence, it is suitable only for simple wave theories like Stokes I or II, provided wave length is given or approximated explicitly. This approach is also rather simplistic, as it only accounts for 0/1 cells. The resulting waves show initial disturbances similar to steps due to this lack of partial cells, and need more time to regularize their profile.

The second approach is the latest effort done, and it was recently presented in Jacobsen et al. (2012). Their approach is complete, accounting for wet, dry and also partial cells. However, they lack active wave absorption at the boundary, as they use relaxation zones. The technique has clear disadvantages, as the existence of a wave damping region is known to produce an increment of the mean water level (Mendez et al., 2001). It also increases the computational domain by approximately two wave lengths (Wei and Kirby, 1995), which is quite inconvenient for already large domains and prototype applications.

The new IHFOAM wave generation BC introduces several new features such as active wave absorption and a specific module to replicate laboratory wavemakers. It has been coded from scratch to realistically generate waves at the boundaries according to a number of wave theories, including: Stokes I, II and V, cnoidal and streamfunction regular waves; Boussinesq solitary wave; irregular (random) waves, first and second order; and piston-type wavemaker velocity profile replication. To choose among wave generation theories it is advised to use the classic graph by Le Méhauté (1976), shown in figure 2.1.

This graph can be easily applied by means of the preprocessing tool located in your **Reference** folder, called **waveTheory**. Running the script will require wave height (H), water depth (h) and wave period (T), and the output is a Le Méhauté (1976) graph with your wave state marked as a point.

As a convention for the wave generation BCs to work appropriately, gravity has to act in the negative direction of the Z axis, and the lowest points of each wave generation boundary must be placed at the same level, but that level can be different from patch to patch. Angles are measured from the X axis, and increasing anticlockwise.

Table 2.1 is presented as a summary of the wave theories that are listed next. It

[1]http://openfoamwiki.net/index.php/Contrib_groovyBC
[2]http://en.wikipedia.org/wiki/File:Water_wave_theories.svg

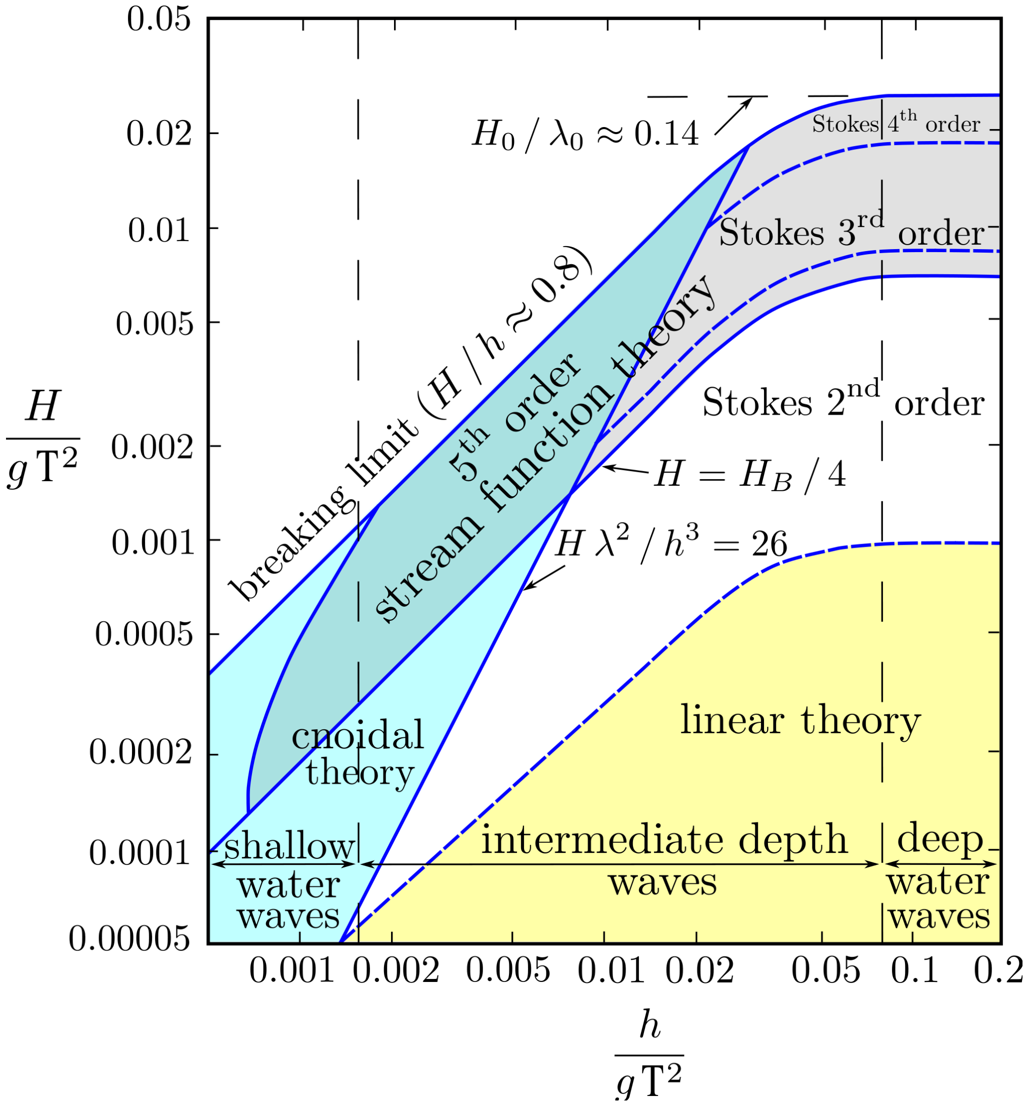


Figure 2.1: Wave theories range of applicability. Le Méhauté (1976). Taken from[2].

Theory	Reference	Comments
Stokes I and II	Dean and Dalrymple (1991)	
Stokes V	Skjelbreia and Hendrickson (1960)	
Cnoidal	Svendsen (2006)	Best fit solver.
Streamfunction	Fenton (1988)	No solver programmed. Its input is the output coefficients from Fenton (1988) program.
Solitary wave	Lee et al. (1982)	Boussinesq theory.

Table 2.1: Wave generation references.

η	free surface elevation
$[u, v, w]$	velocity components
L	wave length
T	wave period
h	water depth
k	wave number
c	wave celerity
ω	angular frequency
ψ	wave phase shift
θ	wave phase
β	wave propagation direction

Table 2.2: Variables used in this chapter.

includes the references used and some remarkable information about the solvers implemented within the boundary condition. Finally, all the common variables used in this part are referenced in table 2.2.

2.2 Wave Theories

2.2.1 Stokes I

Stokes I, Airy wave theory, small amplitude waves or linear waves is the most simple analytical solution for water waves. It was developed by Airy (1845) and still today is one of the most widely used, not only for its ease of implementation but also because it is accurate enough for some engineering approximations. Despite its narrow theoretical range of applicability, the practical use is wider, as this theory can also be further developed and extended to higher order, as presented on section 2.2.2.

The main assumptions of this particular wave theory are as follows:

- Continuous, homogeneous, incompressible and inviscid fluid.
- Coriolis forces are neglected.
- Surface tension is neglected.
- Pressure at the free surface is uniform and constant.
- The flow is irrotational.
- The bottom is fixed and impervious.
- The relative wave height is small ($\frac{H}{h} << 1$).

Waves can be represented by several magnitudes and relations between them. The most important one is the dispersion relation, a transcendental equation to obtain wave length (L) for a certain depth given a period:

$$L = \frac{g\,T^2}{2\pi}\tanh\left(\frac{2\pi h}{L}\right) \tag{2.1}$$

Other important relations are:

$$k = \frac{2\pi}{L} \tag{2.2a}$$
$$\omega = \frac{2\pi}{T} \tag{2.2b}$$
$$c = \frac{L}{T} \tag{2.2c}$$
$$L_0 = \frac{g\,T^2}{2\pi} \tag{2.2d}$$

The solution to this wave theory is based on a potential function, from which free surface elevations and the velocity field can be obtained. For a 2D wave, travelling in the positive direction of X axis the expressions are:

$$\eta = \frac{H}{2}\cos(kx - \omega t + \psi) \tag{2.3}$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(kx - \omega t + \psi) \tag{2.4a}$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(kx - \omega t + \psi) \tag{2.4b}$$

For these expressions to be applicable, the assumption that the lowest coordinate of the boundary is placed in $z = 0$ should be made. Nevertheless this is not general, and some simulations may have wave generating boundaries with different lowest levels. To account for it, we may view this z coordinate as a local one, $z = h^* + z^*$, in which the reference level for z^* is the initial still water level, and h^* is the local water depth in the current boundary.

Taking into account that 2D is a special case within the 3D scope there is a real need to extend this formulation to the general case. This is straightforward, setting the 2D case in the desired direction (β) and projecting the result in the X and Y axes, assuming that gravity always works in the Z axis. This process yields:

$$\eta = \frac{H}{2}\cos(\theta) \tag{2.5}$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta)\cos(\beta) \tag{2.6a}$$

$$v = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta)\sin(\beta) \tag{2.6b}$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(\theta) \tag{2.6c}$$

where $\theta = k_x x + k_y y - \omega t + \psi$. Now $k_x = k\ \cos(\beta)$ is the projected wave number in

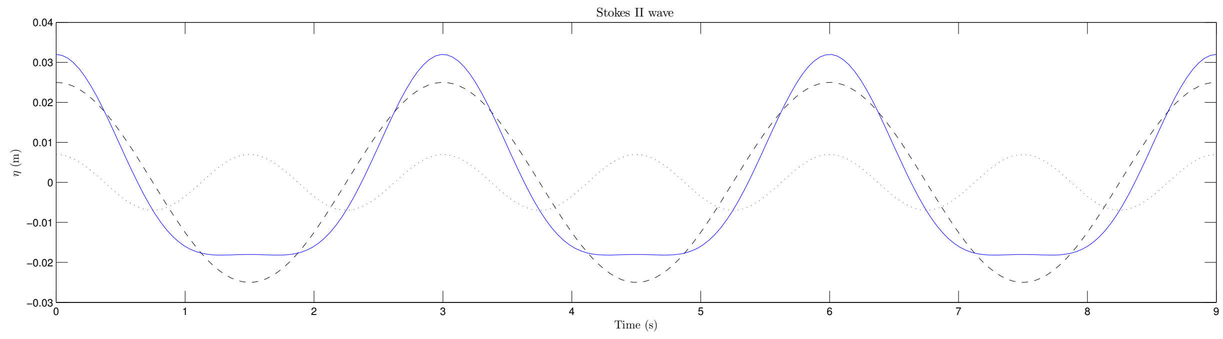


Figure 2.2: Stokes II wave theory: H = 5 cm; h = 40 cm; T = 3 s. Stokes I component is the dashed line, the second order contribution is the dotted line. The final wave is shown in a continuous line. The change in shape of the wave is evident.

the X axis and $k_y = k \sin(\beta)$ is the same for the other horizontal direction. From now on, only the 2D version of the wave theories will be presented, since the transformation to 3D is straightforward.

Should this wave theory be used outside its range, waves are likely to decompose because free surface elevation and the velocity field are not those needed for equilibrium. This causes a secondary wave crest in the trough of the main waves, which never appears in nature.

2.2.2 Stokes II

Stokes II is a further development of Stokes I wave theory, which adds a second order term to the previously studied theory. It is still very easy to implement, as only the dispersion relation has to be iteratively solved. The practical effect of the second term is the sum of another wave which oscillates twice as fast. The effect of the second order can be seen in figure 2.2, in which a 5 cm wave on 40 cm of water and 3 seconds of period is presented.

The resulting expressions for free surface and velocity components are:

$$\eta = \frac{H}{2}\cos(\theta) + k\frac{H^2}{4}\frac{3-\sigma^2}{4\sigma^3}\cos(2\theta) \tag{2.7}$$

$$u = \frac{H}{2}\omega\frac{\cosh(kz)}{\sinh(kh)}\cos(\theta) + \frac{3}{4}\frac{H^2\omega k\cosh(2kz)}{4\sinh^4(kh)}\cos(2\theta) \tag{2.8a}$$

$$w = \frac{H}{2}\omega\frac{\sinh(kz)}{\sinh(kh)}\sin(\theta) + \frac{3}{4}\frac{H^2\omega k\sinh(2kz)}{4\sinh^4(kh)}\sin(2\theta) \tag{2.8b}$$

with $\sigma = \tanh(kh)$.

Once again, if this theory is extended further outside its range of application a secondary wave crest will appear in the wave trough of the primary wave component.

2.2.3 Stokes V

The previous theories represent the smaller waves fairly well. However, as wave height increases they can no longer be considered of small amplitude, therefore another theory is necessary to be able to represent these finite height waves. There are a number of fifth order Stokes waves theories. Based on previous experience regarding wave generation, the theory presented in Skjelbreia and Hendrickson (1960) is used.

For the full development of the theory we refer the reader to the original paper, but the equations to solve and the expressions for free surface and velocity are given here. A system of two transcendental equations with two unknowns must be solved iteratively to obtain wave length (L) and the λ parameter that appear throughout the expressions. These two equations are (2.9) and (2.10).

$$\frac{\pi H}{h} = \frac{L}{h}\left[\lambda + \lambda^3 B_{33} + \lambda^5\left(B_{35} + B_{55}\right)\right] \tag{2.9}$$

$$L = L_0 \tanh\left(\frac{2\pi h}{L}\right)\left(1 + \lambda^2 C_1 + \lambda^4 C_2\right) \tag{2.10}$$

where factors A_{ij}, B_{ij} and C_i depend on L, and have polynomial expressions. In the boundary condition, Newton-Raphson algorithm is used to solve the system. This is not always possible, specially out of range, so if no convergence is obtained the simulation stops and throws a fatal error.

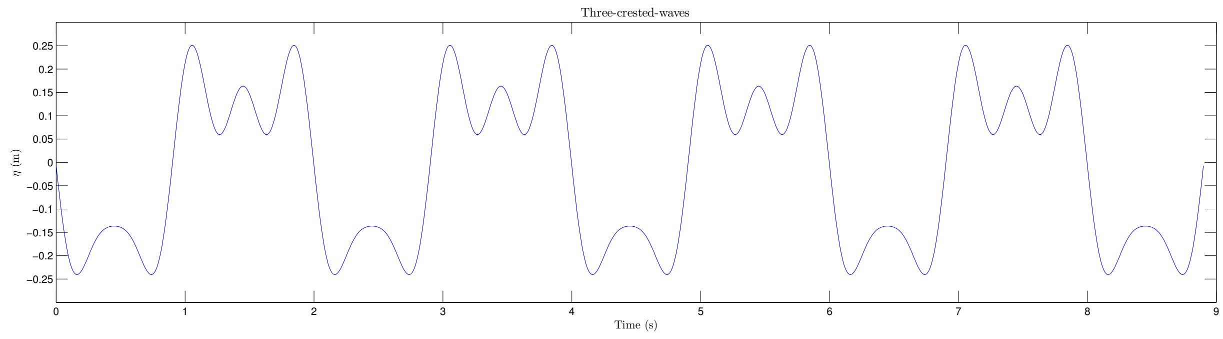


Figure 2.3: Three crested waves Stokes V wave theory: H = 30 cm; h = 40 cm; T = 3 s. Theory has been used out of range, as the wave is very close to breaking. Cnoidal or streamfunction should be used to realistically represent such waves.

In a practical sense there is a way to avoid N-R solver based on the fact that the resulting wave length is always greater than the linear theory one. This yields to a bound for Stokes V wave length, between the mentioned one and L_0. This interval can be divided into a great number of points, and for each of them two different λ values can be easily computed with equation 2.9 and 2.10. The point which has the smallest difference between both values of λ will be the solution. For waves within the range of applicability and using a high resolution for L this approximation works fine. Outside that range the so called three crested waves might appear, as shown in figure 2.3.

The expressions for the boundary condition are:

$$\begin{aligned}\eta =&\lambda \cos(\theta)/k + \left(\lambda^2 B_{22} + \lambda^4 B_{24}\right) \cos(2\theta)/k + \left(\lambda^3 B_{33} + \lambda^5 B_{35}\right) \cos(3\theta)/k \\ &+ \lambda^4 B_{44} \cos(4\theta)/k + \lambda^5 B_{55} \cos(5\theta)/k\end{aligned} \tag{2.11}$$

$$\begin{aligned}u =&a_{vel1} \cosh(kz) \cos(\theta) + a_{vel2} \cosh(2kz) \cos(2\theta) + a_{vel3} \cosh(3kz) \cos(3\theta) \\ &+ a_{vel4} \cosh(4kz) \cos(4\theta) + a_{vel5} \cosh(5kz) \cos(5\theta)\end{aligned} \tag{2.12}$$

$$\begin{aligned}w =&a_{vel1} \sinh(kz) \sin(\theta) + a_{vel2} \sinh(2kz) \sin(2\theta) + a_{vel3} \sinh(3kz) \sin(3\theta) \\ &+ a_{vel4} \sinh(4kz) \sin(4\theta) + a_{vel5} \sinh(5kz) \sin(5\theta)\end{aligned} \tag{2.13}$$

in which the amplitude for each term is:

$$a_{vel1} = \frac{2\pi}{Tk}\left(\lambda A_{11} + \lambda^3 A_{13} + \lambda^5 A_{15}\right) \tag{2.14a}$$

$$a_{vel2} = 2\,\frac{2\pi}{Tk}\left(\lambda^2 A_{22} + \lambda^4 A_{24}\right) \tag{2.14b}$$

$$a_{vel3} = 3\,\frac{2\pi}{Tk}\left(\lambda^3 A_{33} + \lambda^5 A_{35}\right) \tag{2.14c}$$

$$a_{vel4} = 4\,\frac{2\pi}{Tk}\left(\lambda^4 A_{44}\right) \tag{2.14d}$$

$$a_{vel5} = 5\,\frac{2\pi}{Tk}\left(\lambda^5 A_{55}\right) \tag{2.14e}$$

2.2.4 Cnoidal

Cnoidal waves are a type of nonlinear regular waves with a distinctive shape. They are naturally present in nature, when wave length is large compared to the water depth. This leads to a particular and very recognizable wave shape, with very long and flat troughs and steep wave crests. Their starting point is the Korteweg-de Vries equation:

$$\frac{\partial \eta}{\partial t} + \sqrt{gh}\left(1 + \frac{3}{2}\frac{\eta}{h}\right)\frac{\partial \eta}{\partial x} + \frac{h^2}{6}\sqrt{gh}\frac{\partial^3 \eta}{\partial x^3} = 0 \tag{2.15}$$

This theory has two asymptotic limits. First, if wave length tends to infinity, a solitary wave will be obtained. Second, if the relative wave height is small, the results will match Stokes I theory. There are also a number of cnoidal wave theories in the literature, in this case Svendsen (2006) has been used.

Prior to obtaining free surface or velocities, equation 2.16 must be solved to obtain the wave length (L) and the elliptic parameter (m):

$$\frac{c^2}{gh} = 1 + \frac{H}{mh}\left(2 - m - 3\frac{E_m}{K_m}\right) \tag{2.16a}$$

$$\frac{HL^2}{h^3} = \frac{16}{3} m K_m^2 \tag{2.16b}$$

$$c = \frac{L}{T} \tag{2.16c}$$

where K_m is the complete elliptic integral of the first kind and E_m is the complete elliptic integral of the second kind, both dependent on m. Equation 2.16c is used to calculate the error. The elliptical parameter m can take values between 0 and 1. When it reaches the lowest value, this theory is equivalent to Stokes I. And when m approaches 1 the resultant waves resemble a solitary wave in shape. Effectively the range for cnoidal waves presented in Le Méhauté (1976) covers values of m between 0.5 and $0.\bar{9}$, therefore the implemented solver tests all the values in between with a 10^{-5} resolution for the smallest error. It is an assumable cost, since it is only performed once, and expressions are fully explicit then.

When the wave characteristics have been obtained, free surface is calculated using equation 2.17.

$$\eta = H \left[\frac{1}{m} \left(1 - \frac{E_m}{K_m} \right) - 1 + cn^2 \left[2K_m \left(\frac{x}{L} - \frac{t}{T} \right) \Big|_m \right] \right] \tag{2.17}$$

in which cn is Jacobi elliptic function. The expression for velocities involves knowing η in advance, and the ability to calculate several derivatives, which are done numerically. Velocity components are presented in equation 2.18.

$$u = c\,\frac{\eta}{h} - c\left(\frac{\eta^2}{h^2} + \frac{\overline{\eta^2}}{h^2} \right) + \frac{1}{2}\,c\,h \left(\frac{1}{3} - \frac{z^2}{h^2} \right) \eta_{xx} \tag{2.18a}$$

$$w = -\,c\,z \left[\frac{\eta_x}{h} \left(1 - \frac{2\eta}{h} \right) + \frac{1}{6} h \left(1 - \frac{z^2}{h^2} \right) \eta_{xxx} \right] \tag{2.18b}$$

where η_x, η_{xx} and η_{xxx} are the first, second and third derivatives of the free surface elevation with respect to x, respectively; and $\overline{\eta^2}$ is the mean over a wave period of the free surface elevation squared.

2.2.5 Streamfunction Theory

When Stokes V or cnoidal wave theories are not applicable, usually a high order streamfunction solution can be used. And what is more, it can be extended to any order because the computer solves the system of equations to obtain the needed coefficients. This approach allows the generation of waves very near to the breaking condition.

The specific procedure is not explained here; the reader may be referred to Rienecker and Fenton (1981) and Fenton (1988) for the details, but it is based on expressing the complex potential solution in a Fourier series. The solver has not been coded within the boundary condition, instead a program called “Fourier”, developed also by Fenton and similar to the one presented in Fenton (1988) is used. It is coded in C and is currently available for download[3], be sure to download the old version. All it needs as input is the nondimensionalized wave height ($\frac{H}{h}$) and period ($T\sqrt{\frac{g}{h}}$) or wave length, plus the current magnitude, which can be taken as 0 if modelling a wave flume or tank. The output is formed by several wave parameters and two sets of a given number of components: 10 suffice for ordinary waves, but up to 32 can be computed for steeper ones. These, along with wave length and the mean fluid speed in the frame reference of the wave (also provided by the program) yield to the free surface elevation and velocity field, therefore they constitute the input for the boundary condition.

Free surface is calculated using equation 2.19.

$$\eta = h \sum_{j=1}^{N} E_j \cos\left[j\left(k\,x - \omega\,t\right)\right] \tag{2.19}$$

where E_j are one set of the given coefficients. The velocity components are presented in equation 2.20.

$$u = c - \bar{U} + \sqrt{g h^3 k^2} \sum_{j=1}^{N} j B_j \frac{\cosh\left(jkz\right)}{\cosh\left(jkh\right)} \cos\left[j\left(k\,x - \omega\,t\right)\right] \tag{2.20a}$$

$$w = \sqrt{g h^3 k^2} \sum_{j=1}^{N} j B_j \frac{\sinh\left(jkz\right)}{\cosh\left(jkh\right)} \sin\left[j\left(k\,x - \omega\,t\right)\right] \tag{2.20b}$$

in which $\bar{U}$ is the previously mentioned mean fluid speed in the frame reference of the wave and B_j is the other set of coefficients.

[3] http://johndfenton.com/Steady-waves/Fourier.html

2.2.6 Solitary Wave

From the Korteweg-de Vries equation some solutions with permanent form can be obtained; this means that their shape does not change during propagation. The first of these solutions is the solitary wave, which is not an oscillatory wave, but a translational wave. What that means is that all the particles of the wave move in the direction of propagation, because its shape is always over the still water level, without evident wave troughs.

There are a number of solitary wave theories. Some of them are reviewed in Lee et al. (1982), from which the expressions for velocities and free surface can been taken. Boussinesq theory is the chosen one, but any of them would be easily implemented and added.

The free surface expression is as follows:

$$\eta = H \,\mathrm{sech}^2 \left[\sqrt{\frac{3H}{4h^3}} X \right] \tag{2.21}$$

in which $X = (x - c\,t)$, and the wave speed c is $\sqrt{g\,(h+H)}$.

The velocity components are straightforward, although they involve derivatives in η. They are presented in equation 2.22.

$$\frac{u}{\sqrt{gh}} = \frac{\eta}{h} \left[1 - \frac{1}{4}\frac{\eta}{h} + \frac{h}{3}\frac{h}{\eta} \left(1 - \frac{3}{2}\frac{z^2}{h^2} \right) \frac{d^2\,\eta}{d\,X^2} \right] \tag{2.22a}$$

$$\frac{w}{\sqrt{gh}} = \frac{-z}{h} \left[\left(1 - \frac{1}{2}\frac{\eta}{h} \right) \frac{d\,\eta}{d\,X} + \frac{1}{3} h^2 \left(1 - \frac{1}{2}\frac{z^2}{h^2} \right) \frac{d^3\,\eta}{d\,X^3} \right] \tag{2.22b}$$

Using the presented equations and assuming the boundary to be at $x = 0$, the free surface will start at the highest point of the solitary wave. In order to generate the full wave an artificial lag in space has to be added. There is an obvious problem with this, as a solitary wave theoretical wave length is infinite. This lag in space is directly translated in a lag in time, therefore there is a need to keep it as low as possible to reduce the simulation (and computational) time. Nevertheless the free surface decreases rapidly and an effective wave length can easily be defined. The usual criterion to calculate it is to set a percentage of the maximum wave height in which the simulation will start. For example if 1% is fine, the lag will be $l = \frac{3.5}{\sqrt{\frac{H}{h}}}$. Sometimes 5% is preferred, in this case the lag will be $l = \frac{2.5}{\sqrt{\frac{H}{h}}}$. Here the 1% criterion is used because the other one would involve half a cell of error in the case in which wave height is discretized using 10 cells.

2.2.7 Irregular Waves

Waves in nature are seldom purely regular. For example, swell waves are more or less regular, but they also appear mixed with sea waves. In order to simulate real conditions, an irregular wave generating theory is needed. Most of the times a first order theory will suffice to represent the sea state, while other times second order interactions between first order components will be needed to get more accurate results.

2.2.7.1 First Order

First order directional irregular waves are generated as a linear superposition of Stokes I waves for a given number of components (N), see equation 2.23. This approach is physically correct, as most of the times by discretizing a real wave spectrum with a large number of components, very small amplitudes are obtained. Each of the components is defined by its wave height (H_i), wave period (T_i), wave phase (ψ_i) and the direction of propagation (β_i). Free surface and orbital velocity components are given by:

$$\eta = \sum_{i=1}^{N} \frac{H_i}{2} \cos\left(k_{x\,i}\, x + k_{y\,i}\, y - \omega_i t + \psi_i\right). \tag{2.23}$$

$$u = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\Delta\beta_i) \frac{\cosh(k_i z)}{\sinh(k_i h)} \cos(\theta_i) \cos(\beta_i) \tag{2.24a}$$

$$v = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\Delta\beta_i) \frac{\cosh(k_i z)}{\sinh(k_i h)} \cos(\theta_i) \sin(\beta_i) \tag{2.24b}$$

$$w = \sum_{i=1}^{N} \frac{H_i}{2} \omega_i \cos^2(\Delta\beta_i) \frac{\sinh(k_i z)}{\sinh(k_i h)} \sin(\theta_i) \tag{2.24c}$$

where $\theta_i = k_{ix}x + k_{iy}y - \omega_i t + \psi_i$. $\Delta\beta_i$ corresponds to the difference between each

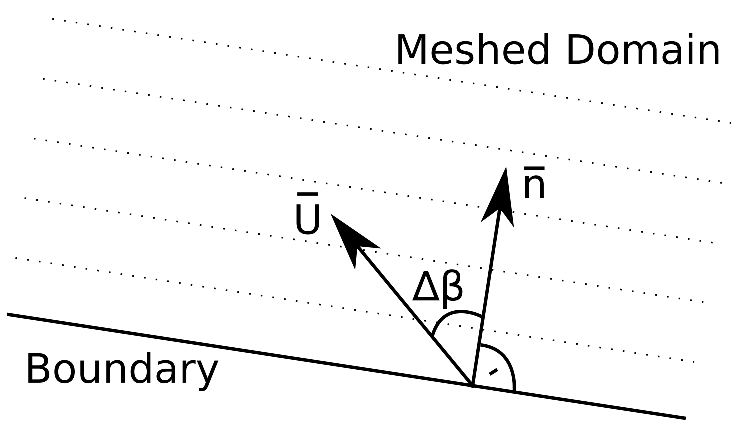


Figure 2.4: $\Delta\beta$ definition.

component direction and the vector normal to the boundary pointing into the domain, as shown in figure 2.4.

Despite the directionality, the boundaries are not capable of generating outward ($|\Delta\beta| > \pi/2$) or tangential ($|\Delta\beta| = \pi/2$) waves, so these should be left out, taking only angles smaller than $\pi/2$ for each side. For this purpose a classical square cosine function ($f = \cos^2(\Delta\beta)$) has been used. The resultant decreasing factor (f), as a function of the angle difference is shown in figure 2.5. This factor is multiplied by the velocities of each component at the boundary, as presented in equation 2.24.

2.2.7.2 Second Order

Since with first order wave generation theory the group bound wave is not reproduced (Barthel et al., 1983), a correct wave representation needs to include second order effects. The absence of such a feature causes spurious free waves to be generated. Therefore, long wave effects are underestimated in shallow water and overestimated in deep water as stated in Sand (1982). Second order irregular wave generation takes into account the interaction between the individual primary wave components, two by two, and it is built on top of the first order method. Currently, it only supports components travelling in the same direction, therefore $\beta_i = \text{const}$.

In this case, the theory by Longuet-Higgins and Stewart (1960) completed with Baldock et al. (1996) is applied as in Torres-Freyermuth et al. (2010). The reader is referred to these papers for further details, as only a brief description will be given here. The second order effects are added to the first order free surface level (equation 2.25) and to

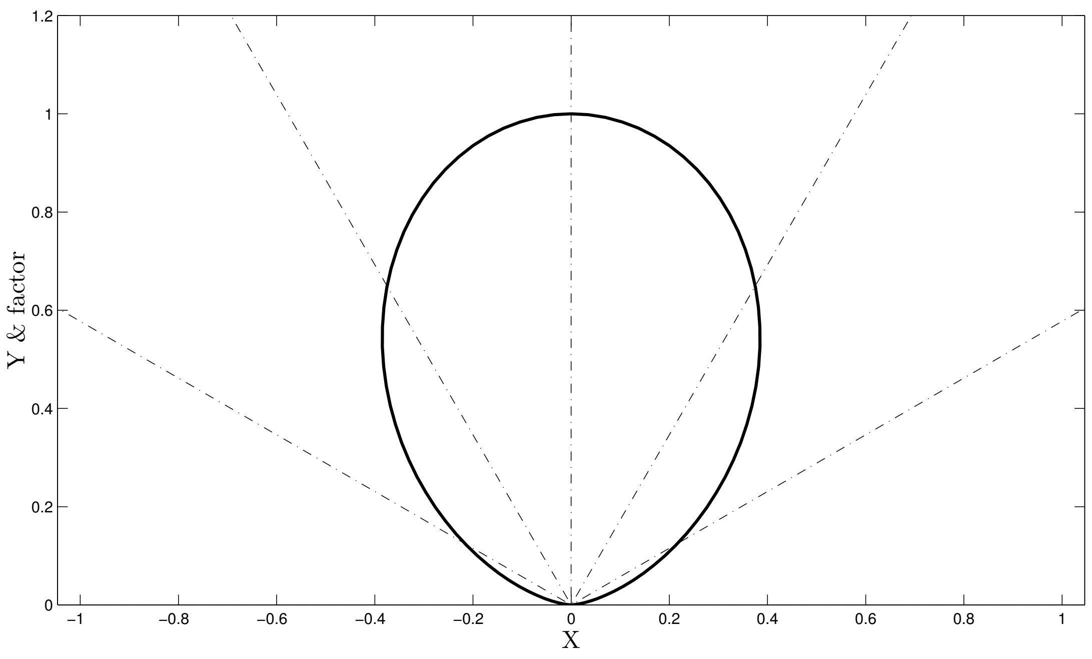


Figure 2.5: Square cosine decreasing factor function presented in bold continuous line. The X and Y axes have the same scale in order to evaluate the angle $\Delta\beta$. Only for this figure the angle reference is axis Y. The discontinuous lines are $\Delta\beta$ rays each 30°. The reduction factor is the ordinate of the cut point between the ray and the function. Angles greater than 90°and smaller than -90°have a decreasing factor equal to zero.

the velocity. To account for all the velocity components simultaneously the final velocity potential (ϕ) expression is given here in equation 2.26.

$$\eta = \sum_{i=1}^{N} \eta_i + \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} \frac{H_n H_m}{8g} [C \cos(\theta_n - \theta_m) - D \cos(\theta_n + \theta_m)] \tag{2.25}$$

$$\begin{aligned} \phi = \sum_{i=1}^{N} \phi_i + \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} \\ \Bigg[\frac{E \cosh[(k_n - k_m)(h + z^*)] \sin(\theta_n - \theta_m)}{g(k_n - k_m) \sinh[(k_n - k_m)h] - (\omega_n - \omega_m)^2 \cosh[(k_n - k_m)h]} \\ - \frac{F \cosh[(k_n + k_m)(h + z^*)] \sin(\theta_n + \theta_m)}{g(k_n + k_m) \sinh[(k_n + k_m)h] - (\omega_n + \omega_m)^2 \cosh[(k_n + k_m)h]} \Bigg] \end{aligned} \tag{2.26}$$

The only new variables introduced are C, D, E and F, which are defined in Longuet-Higgins and Stewart (1960), which are given in equation 2.27. C and E control the subharmonic generation while D and F generate the superharmonic interaction.

$$C = \frac{\left[2\omega_1\omega_2\left(\omega_1-\omega_2\right)\left(\alpha_1\alpha_2+1\right)+\omega_1^3\left(\alpha_1^2-1\right)-\omega_2^3\left(\alpha_2^2-1\right)\right]\left(\omega_1-\omega_2\right)\left(\alpha_1\alpha_2-1\right)}{\omega_1^2\left(\alpha_1^2-1\right)-2\omega_1\omega_2\left(\alpha_1\alpha_2-1\right)+\omega_2^2\left(\alpha_2^2-1\right)} + \left(\omega_1^2+\omega_2^2\right)-\omega_1\omega_2\left(\alpha_1\alpha_2+1\right) \tag{2.27a}$$

$$D = \frac{\left[2\omega_1\omega_2\left(\omega_1+\omega_2\right)\left(\alpha_1\alpha_2-1\right)+\omega_1^3\left(\alpha_1^2-1\right)+\omega_2^3\left(\alpha_2^2-1\right)\right]\left(\omega_1+\omega_2\right)\left(\alpha_1\alpha_2+1\right)}{\omega_1^2\left(\alpha_1^2-1\right)-2\omega_1\omega_2\left(\alpha_1\alpha_2+1\right)+\omega_2^2\left(\alpha_2^2-1\right)} - \left(\omega_1^2+\omega_2^2\right)+\omega_1\omega_2\left(\alpha_1\alpha_2-1\right) \tag{2.27b}$$

$$E = -\frac{1}{8}H_1H_2\left[2\omega_1\omega_2\left(\omega_1-\omega_2\right)\left(1+\alpha_1\alpha_2\right)+\omega_1^3\left(\alpha_1^2-1\right)-\omega_2^3\left(\alpha_2^2-1\right)\right] \tag{2.27c}$$

$$F = -\frac{1}{8}H_1H_2\left[2\omega_1\omega_2\left(\omega_1+\omega_2\right)\left(1-\alpha_1\alpha_2\right)-\omega_1^3\left(\alpha_1^2-1\right)-\omega_2^3\left(\alpha_2^2-1\right)\right] \tag{2.27d}$$

2.2.8 Piston Wavemaker Emulation

One of the goals of a numerical model is to replicate natural processes. In coastal engineering it is usually very difficult to take field measurements due to high energetic conditions. To effectively study such phenomena there is a need for experimental facilities. In such facilities, waves are generated by moving elements within the water. A great variety of wavemakers exist, but the most widespread are piston-type, which move back and forth in the horizontal direction. They generate a constant velocity profile all along the water column. Since this profile generates a wave whose equilibrium profile is different (except if the wave is a linear long wave) some spurious waves called evanescent modes are generated near the wavemaker, but this effect soon vanishes as the wave propagates away.

There are 4 main cases for this boundary condition depending on the data available, but those get reduced to only 2 cases at the first time step of the simulation.

The first case (**tx**) provides a series of time and displacement of the wavemaker, which is the most usual output of experimental facilities. From these, velocity of the wavemaker is calculated as a first order forward derivative, as shown in equation 2.28.

$$U = \frac{X_{i+1} - X_i}{t_{i+1} - t_i} \tag{2.28}$$

in which for a given time t: $t_i \leq t$ and $t_{i+1} > t$.

This expression is quite convenient because it does not require a homogeneous sampling rate. The result is the second case (**tv**), when a time series of velocities is provided.

The third and fourth cases are the same as before, but an additional series of free surface elevation at the wavemaker is also specified (**txeta, tveta**). Not all the experimental wavemakers are capable of providing such feedback, but it allows us to trigger the active wave absorption without any further assumptions. To prescribe the free surface level, the temporal series is interpolated linearly.

2.3 Numerical Implementation

Numerical implementation is common to all wave theories. Only a small change is made in the case of the piston-type wavemaker replication when no free surface is provided. The current approach takes into account three types of cells: wet cells, which have all the vertices below the free surface level; dry cells, which have all the vertices above the free surface level; and partial cells, in which the free surface is between the lowest and highest vertices of the cell.

Wave generation involves setting the values of velocity and VOF function (field "alpha1", α_1 from now on), therefore their implementation is separate, but they share most of the source code. Pressure is not set directly, it is calculated using the "buoyantPressure" boundary condition. This special function available in OpenFOAM® calculates the normal gradient from the local density gradient. This ensures that the second derivative of pressure in the orthogonal direction to the boundary is zero.

During the first time step of the model, and only then, several processes are carried out. Wave generation variables are read and relevant variables are calculated using wave theory. Also the initial still water depth at the patch is measured, as the wet area of the patch (sum of the individual face areas times α_1) over the total patch area.

For the piston-type wavemaker replication, if free surface is not provided, the corresponding constant velocity profile along the whole water column is applied. This is done by multiplying such velocity by α_1 in each cell, in order not to introduce air velocity. No further considerations are made. If the free surface elevation is provided, it is interpolated linearly in time and the procedure is as in the general case, which is explained next.

For the rest of the cases presented, an expression to calculate the free surface is either provided or calculated by means of the corresponding wave theory. Consequently, at each time step, the theoretical and measured free surface levels can be compared in order to trigger active wave absorption. The following method takes into account the possibility of high amplitude reflected waves reaching the generation patch, and makes the simulation more stable.

In the next lines "zero gradient" is used to indicate that the boundary face value is set to the value of its owner cell or mathematically for any variable q: $\frac{\partial q}{\partial x_i} n = 0$, where n is the normal direction to the face. Please, note that it does not refer to "zeroGradient" OpenFOAM®built in BC.

There are three different areas, which are indicated in figure 2.6, depending on whether the measured level is higher than the theoretical one (positive reflected wave, left panel on the figure) or lower (negative reflected wave, right panel). This disposition provides three different areas ("a", "b" and "c") and two interfaces between them. The implementation is explained next, and can be summarized as shown in Table 2.3.

Zone "a" is common in both cases, and corresponds to air ($\alpha_1 = 0$). Therefore, the faces within it have α_1 and velocity set to zero.

The interface between "a" and "b" is different depending on the case, because wave velocity is set only below the theoretical level. When M > T carries a zero gradient value for α_1 and zero velocity. If T > M, a special procedure is performed. The intersection between the theoretical water level and the cell is calculated to obtain the corresponding α_1 of the cell. Also the centroid of the cell wet part is obtained to calculate the velocity at that point. Velocity is set as the multiplication of both quantities. This is done to prevent spurious air velocities in the interface, which tend to lower the Courant number and to lower stability.

Zone "b" also depends on the case. When M > T, α_1 is set to zero gradient and

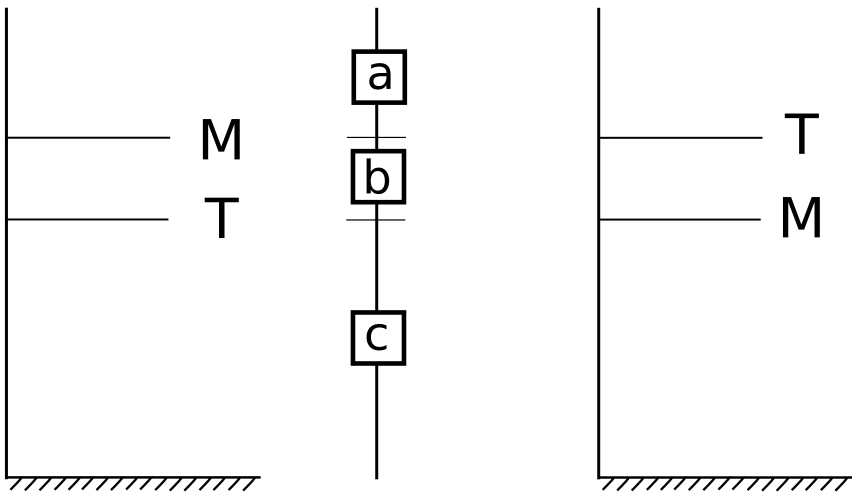


Figure 2.6: Water level in a wave generating patch. “M” stands for measured value, “T” stands for theoretical value. Three zones (a, b and c) and two interfaces are present in each case.

velocity is directly zero. When T > M, α_1 is set to 1 when water flux is inwards and to zero gradient otherwise. Velocity is set to the theoretical value. This sometimes may cause water droplets to appear in the boundary, especially when the reflected amplitude is large (on the order of several cells).

The interface between “b” and “c” sometimes has water above (M > T) and sometimes should (T > M), but due to the reflected waves it does not. This condition changes its behaviour depending on this situation, as shown in Table 2.3 (left vs right table). In the first case, α_1 is set to zero gradient, and the velocity is calculated in the same way as in zone “a-b” for T > M, for the same reasons explained above. In the second case if water is flowing in, it is set to one, and to zero gradient otherwise. Velocity in this case is set to the calculated value.

Finally, the zone “c” is also common to both cases, as it is always under the measured and theoretical values, therefore it corresponds to water ($\alpha_1 = 1$). If the flow is entering the domain, α_1 on the face is set to one, otherwise it is set to zero gradient. This configuration is more stable than setting it always to one, as it has been observed that this other solution can more easily cause α_1 to reach negative values despite the fact that the “MULES” solver ensures boundedness between zero and one. Velocities are set to the theoretical value.

When an interface coincides with a cell, that cell is automatically an interface cell. Should both of the mentioned interfaces coincide within a cell (e.g. small reflected waves

	M > T		T > M	
	α_1	U	α_1	U
a	0	0	0	0
a-b	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	0	$\alpha_{1\,calc}$	$U \cdot \alpha_{1\,calc}$
b	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	0	In $\rightarrow 1$ Out $\rightarrow \frac{\partial \alpha_1}{\partial x_i} n = 0$	U
b-c	$\frac{\partial \alpha_1}{\partial x_i} n = 0$	$U \cdot \alpha_{1\,calc}$	In $\rightarrow 1$ Out $\rightarrow \frac{\partial \alpha_1}{\partial x_i} n = 0$	U
c	In $\rightarrow 1$ Out $\rightarrow \frac{\partial \alpha_1}{\partial x_i} n = 0$	U	In $\rightarrow 1$ Out $\rightarrow \frac{\partial \alpha_1}{\partial x_i} n = 0$	U

Table 2.3: Overview of the boundary condition values depending on the zones established in figure 2.6.

or even T = M), the priority of the cell is from below to the top, from “c” to “a”.

The velocity and α_1 can be set in different ways, as decided by the user. The first and most obvious way is face by face. Each of them owns a cell which is formed by a number of points, and by a centroid, all of them having different coordinates. The belonging zone of the face (“a”-“c”) is checked using the highest and lowest points of the cell in the Z direction, while velocity is calculated using the centroid coordinate of the face. This boundary condition also supports an automatic division of the generating patch in vertical slices. Such zones resemble the disposition of individual paddles within a wavemaker, and behave in the precise way to allow closer replication of such a device without the inconvenience of having to manually divide the boundary in advance. Cells within each of these zones lose their x and y coordinates in favour of the centroid of the paddle, but maintain their height, for α_1 and velocity calculation.

Additionally, if active absorption is connected, the face velocities get corrected adding the calculated value to the ones above, in the manner explained in the following section.

Chapter 3

Wave Absorption Boundary Conditions

Active absorption of waves is one of the key features of physical and numerical experiments in coastal engineering. At prototype scale the waves can travel away from the study zone. However, in physical or numerical experiments this is not the case, the domains are either constrained in dimensions, such as in wave basins and flumes, or cannot be placed at an infinite distance because of computational restrictions. This situation causes inconvenient reflections that, if not handled adequately, could influence the experiment by distorting its results. A thorough validation of the boundary conditions is presented in Higuera et al. (2013a) and Higuera et al. (2013b).

3.1 Theory

The first approach for dissipating the outgoing waves in the past has been using passive wave absorbers. Under this classification we can include dissipative beaches, porous media or porous plates and artificial numerical damping. The absorption achieved is not perfect. For example, long waves get reflected even on dissipative beaches, since they do not break, resulting in weakly reflecting BCs. Numerically, passive wave absorbers can be replicated as porous media or as relaxation zones. Practical applications for RANS codes can be found in Pengzhi and Liu (1999), Lara et al. (2006) and for Boussinesq-type models in Wei and Kirby (1995) or Losada et al. (2008).

Passive absorbers have already been developed in OpenFOAM® and were recently presented in Jacobsen et al. (2012). They use relaxation zones. The technique has clear disadvantages as the existence of a wave damping region is known to produce an increment of the mean water level (Mendez et al., 2001). It also increases the computational domain by around two wave lengths (Wei and Kirby, 1995), which is quite inconvenient for already large domains and prototype applications.

The second method is active wave absorption. It was first developed for experimental facilities. The system is based on modifying the wavemaker movement based on a measured magnitude (feedback) so that it continues to generate the target wave, while preventing the re-reflection of incoming waves. This system can be applied to numerical models. In this context, most of the times the boundaries are fixed, and as a result wave absorption is achieved by imposing the correct velocity profile on it.

Active wave absorption systems can be divided into three categories: 2D, Quasi-3D and 3D. The details for each one are presented as follows.

3.1.1 2D Absorption

The 2D active absorption method is developed as appears in Schäffer and Klopman (2000). It is the easiest technique to be implemented considering the fact that the adjustment for the digital filter is immediate, as it is based on linear shallow water theory. Previous works on other numerical models (Torres-Freyermuth et al., 2010) (Lara et al., 2011) have shown that it works relatively well even when used for waves outside the shallow water range.

It is very convenient to use shallow water theory because the velocity along the water column height is constant, which matches the generation with a piston-type wavemaker. This also makes the evanescent modes cancel out because the velocity profile is the exact one for the progressive wave component. From this wave theory, equation 3.1 can be derived.

$$U\,h = c\,\eta \tag{3.1}$$

where U is the horizontal vertically–integrated (uniform) velocity and c is the wave celerity.

U is the variable to solve for, and h and η are measurements, consequently there is a need to estimate wave celerity (c). This magnitude is dependent on the relative depth of the waves (kh) in the following way:

$$c = \sqrt{g\,h}\sqrt{\frac{\tanh(k\,h)}{k\,h}}. \tag{3.2}$$

Wave number (k) is very difficult to estimate from measurements, so the practical application of equation (3.2) is achieved by a digital filter assimilation. The use of digital filters for active wave absorption is widely used, for further information refer to Christensen and Frigaard (1994) and Troch and De Rouck (1999). More recently, in Wellens (2012), equation (3.3) is used as a rational approximation to equation (3.2).

$$c^* = \sqrt{g\,h}\,\frac{a_0 + a_1(k\,h)^2}{1 + b_1(k\,h)^2}. \tag{3.3}$$

A thorough work to adjust the a_i and b_i coefficients has also been carried out in Wellens (2012).

We apply shallow water regime, as explained: $c = \sqrt{g\,h}$, that yields $a_0 = 1$, $a_1 = 0$ and $b_1 = 0$.

In order to cancel out the reflected waves, the boundary must generate a velocity equal to the incident one but in the other direction. Arranging equation (3.1) so that the free surface corresponds to the reflected one η_R (the one to cancel out) leads to the active wave absorption expression presented in equation (3.4).

$$U_c = -\sqrt{\frac{g}{h}}\,\eta_R \tag{3.4}$$

in which U_c is the correction velocity that is applied to a vector perpendicular to the boundary and pointing into the domain; and the reflected wave height (η_R) is calculated by subtracting the measured elevation at the wavemaker (η_M) from the target one (η_T), according to the expected reflection-free wave generation: $\eta_R = \eta_M - \eta_T$.

This theory was first developed for wave flumes, where the results are usually two dimensional, but can easily be extended to three dimensions. If reflected waves propagate parallel to the wavemaker, the expected behaviour is exactly the same as in the 2D case. If that is not the case, the absorption theory can be applied to the individual paddles of the wavemaker independently.

There is a problem, however, in absorbing in 3D with the 2D theory, as only the wave component perpendicular to the boundary can be absorbed. The other component, which is tangential to the wavemaker, continues to propagate along the boundary until it reaches a lateral dissipative device, if one is available. Otherwise it will reflect, as a plane wavemaker cannot absorb such a wave, because it can only introduce shear stresses. The influence of the latter wave component perturbs the boundary condition, as presented later.

3.1.2 Quasi-3D Absorption

Following the example presented in Schäffer and Klopman (2000) a method to absorb oblique waves is presented. This is only a correction of the already presented 2D absorption theory, enhanced by accounting for a known angle of incidence. The practical application is to reduce velocity by a factor $cos(\Delta\beta)$, as presented in equation (3.5).

$$U_c = -\cos(\Delta\beta)\sqrt{\frac{g}{h}}\,\eta_R \tag{3.5}$$

Note that when incidence is parallel to the boundary ($\Delta\beta = 0$), absorption velocity remains completely unaffected, since it is a true 2D condition. As waves approach the parallel direction, the theory is expected to underperform, never being able to absorb the tangential component of the wave as $\Delta\beta$ approaches $90°$. This will ultimately lead to a stationary wave along the wavemaker if the situation is not handled correctly.

This method was developed for piston wavemakers, but it is modified here taking advantage of the numerical model capabilities, to obtain better performance. Instead of reducing the correction velocity and applying it to the direction perpendicular to the boundary, the correction velocity is still the one calculated with equation (3.4), but this time it is applied to the desired direction. Generally, this new approach is able to absorb better than projecting the velocity and allowing the tangential component to flow along the boundary.

The performance of this boundary condition is outstanding, but there is a clear drawback: most of the times the direction of the incident waves cannot be anticipated, or radiation from the structure causes that direction to change either in time or along the boundary extents. Nevertheless, good performance is expected even for small deviations in the direction of the absorption (Schäffer and Klopman, 2000).

3.1.3 3D Absorption

Although both the 2D and Quasi-3D absorption theories can be used with reasonable results in 3D cases, a specific full 3D theory is needed to obtain lower reflection coefficients or to avoid wave radiation from the absorbing boundaries. The distinctive element of this method is that it uses the feedback additionally to evaluate wave directionality.

The following method is original and quite simple, furthermore it needs no tuning, but due to the complexity of measurements needed, it can only be applied to numerical models. Traditional 3D absorption theories for laboratory wavemakers rely on measuring free surface in front of the individual paddles and by calculating the derivatives or applying a digital filter to this data the directionality is measured. This involves interconnection between wave paddles. Our method eliminates the error introduced by the discrete calculation of derivatives, acting for each paddle independently.

As in the aforementioned methods, this one is also based in shallow water wave theory, i.e. such waves have a velocity field which is constant along the whole water column. As a result, averaging the horizontal components of velocity all over the water depth will theoretically yield the same value as in each point of a vertical line. The same principle is applied to the direction of the horizontal velocity.

The practical application involves the calculation of a mean horizontal velocity with its mean direction for each vertical slice of the boundary. This velocity can be decomposed into two independent components: one normal to the paddle and another tangential to it. Also, the measured free surface level at that vertical slice is needed. Wave directionality cannot be inferred using these two components of velocity at the same time, since

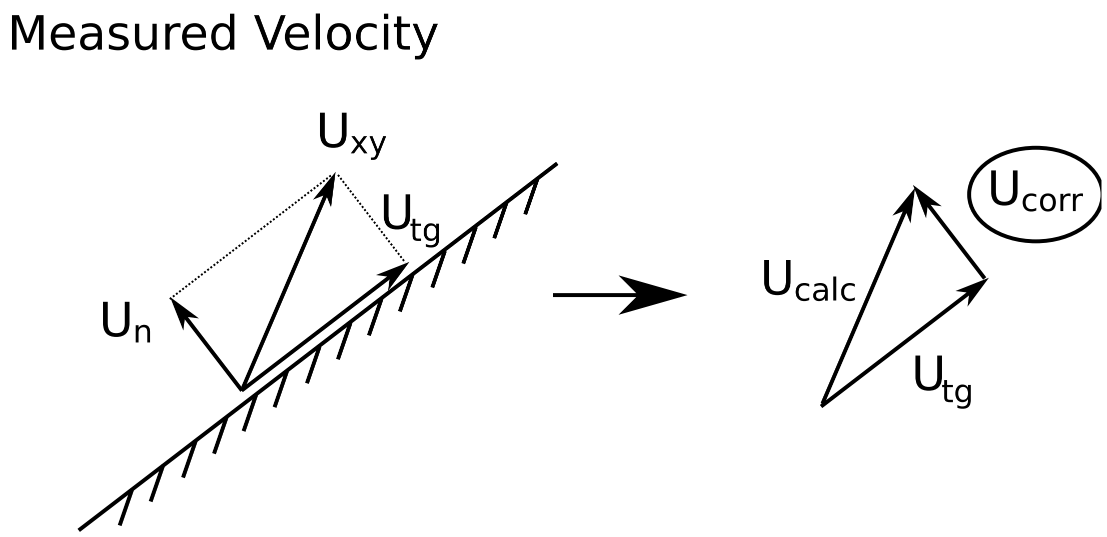


Figure 3.1: Scheme for the full 3D absorption theory.

absorbing waves involves imposing a certain velocity on the boundary, which will completely distort the measurements. Hence, correction velocity can only be prescribed on the perpendicular direction to the paddle, as in the usual wave absorption theories using piston wavemakers, leaving the other component unmodified so that it can be measured. The implementation is presented graphically in figure 3.1.

Using simple calculations wave directionality can then be obtained, as follows:

$$|U_{calc}| = \sqrt{\frac{g}{h}}\,\eta_R. \tag{3.6}$$

where $|U_{calc}|$ is the total expected modulus of velocity, based on the measured free surface level minus the expected one. By decomposing it into the two horizontal components it is obvious that:

$$U_{calc}^2 = U_{corr}^2 + U_{tg}^2. \tag{3.7}$$

in which everything is known except for U_{corr}, the correction velocity that has to be applied in the perpendicular direction to the paddle in order to absorb the waves. To solve for U_{corr} modulus, the square root of a subtraction must be obtained:

$$U_{corr} = \sqrt{U_{calc}^2 - U_{tg}^2}. \tag{3.8}$$

It is easy to notice that when the tangential component of the velocity (U_{tg}) is greater than the total one there will be no real solution. To avoid the imaginary solution, if the value inside the square root is negative it is discarded and taken to be zero.

Once again the problem of having a wave which propagates along the wavemaker is present. There are also two main problems related with corners. The first one is dealing with this tangential wave component which cannot be absorbed along the wave maker. The second is that if we had a 90°corner for two absorbing boundaries, the perpendicular component for the first will be the tangential one for the second and vice versa. This causes a stability problem because wave directionality cannot be correctly estimated. A unique solution solves all these problems. This is to choose a number of paddles near the edges only capable of extracting water according to equation (3.4). The reason for limiting the in-flux of water in such parts is because in the case of an acute corner, water is pushed directly into a small space, causing the level to increase very rapidly, and most of the times reaching the top of the mesh, which will lead to loss of mass in that zone.

3.2 Numerical Implementation

The implementation of pure active wave absorption is easier, as you only need to prescribe the velocities. Pressure boundary condition is set to "buoyantPressure" again, and α_1 to "zeroGradient".

Active wave absorption acting on any boundary works in the same way, including wave generation boundaries. This was shown in figure 1.1 (bottom left part), in which the correction velocity is added to the existing one: either the calculated velocity required by the wave theory, or 0 in case of a purely absorbent patch. Currently only 2D absorption theory is applicable to wave generating boundaries.

The practical application of the absorption theory consists of dividing the boundary into a given number of vertical elements. The minimum is one, a case in which the whole boundary will absorb globally. For each of the individual elements, the initial still water level is calculated and saved. Then each time step the actual water level at each paddle is obtained and the correction velocity is calculated according to the previous theories e.g. using equation (3.4) for 2D.

The correction velocity is then recalculated cell by cell, multiplying the obtained value (checking the paddle in which it is included) by the value of α_1 in that cell. This is done to prevent the propagation of air pockets close to the boundary. The resultant velocity is applied in the perpendicular direction to the face (for 2D and full 3D absorption) or to the given direction (Quasi-3D). If such velocity is positive (in-flux), only the cells below the measured ("M") water level are given this value, the rest are set to zero. Otherwise (out-flux) all the cells are set to the recalculated velocity value. The main reason for doing this is to gain stability, as if a water droplet from a splash was higher than the water level on the patch, it will not propagate when water is flowing into the domain. Otherwise, and it will flow out.

Chapter 4

Usage of the Boundary Conditions

4.1 Overview

In this chapter the new boundary conditions are introduced from a practical point of view.

4.2 Wave Generation Plus Active Absorption Settings

In this section we are going to see the different parameters needed to generate and absorb waves at the same time. The wave generation boundary conditions are controlled by a dictionary file, which has to be located in the **constant** folder. By default, the boundary condition will look for **IHWavesDict**, but it can be changed. This is very convenient when you have to replicate the movement of **n** paddles (i.e. directional wave basin) in which each of them has a different dictionary name, as they move independently.

patchName
```
{
     type             IH_Waves_Inlet(Alpha|Velocity);
     waveDictName     IHWavesDictName;
     value            uniform 0|(0 0 0);
}
```

Within **IHWavesDict** the entries can have any order, and as with all the OpenFOAM® files, you can comment lines by using // to ignore what follows until the end of the line. Additional entries which are not expected can be included, however they are never read. As we will see, several values are needed, and their not being defined will lead to a self-explanatory error at runtime. The same will happen with non-physical values (e.g. negative wave height). Not defined parameters will just take their default value. To avoid any confusion you are advised to always include all your parameters in your dictionary, so that even if the default values change in further versions, you can still reproduce your case.

For full practical reference on how to set the values for any of these parameters, please refer to the appendix A.2.

waveType

The first parameter is the type of the waves. In it we can choose from among the following:

- **regular** (R)
- **irregular** (I)
- **solitary** (S)
- **wavemaker** (W)

The letter inside the parentheses is just for internal reference in this chapter. As it has already been explained, the **irregular** wave case is just a linear sum of Stokes I wave components, while **wavemaker** is the custom-made wavemaker constant velocity profile replication.

This is the main switch which controls the behaviour of the boundary condition, as it can be seen clearly in the source code.

waveTheory - (R||S||W)

This parameter controls the type of waves that are generated. The user should carefully decide which to use based on the specific parameters of the case, making use of the existing tools (i.e. Le Méhauté (1976)). No **waveTheory** is used/read for the **irregular** wave type, because **StokesI** is used, but the use of other theories can easily be implemented as well.

(R)

As presented in chapter 2 the valid theories are:

- **StokesI**
- **StokesII**
- **StokesV**
- **cnoidal**
- **streamFunction**

(S)

The only valid theory for solitary wave generation is:

- **Boussinesq**

Adding new theories is trivial, as it involves coding the formulas and adding a couple of **else if** loops to the existing ones.

(W)

Several other options are available for **wavemaker**, depending on the data you provide:

- **tx**
- **tv**
- **txeta**
- **tveta**

All of them involve a time series (**t** in the names), apart from which you can provide the paddle displacement (**x**), the paddle velocity (**v**), and the free surface elevation at the paddle (**eta**), which allows for a more accurate wave absorption. All of these are explained later.

Regardless of your data **waveTheory** will change to **tveta** the first time the boundary condition is used, as previously explained, and an additional entry named **waveTheory-Orig** along with **v** and **eta** (estimated, if not previously available) will be created in your dictionary file, to keep track of the changes.

Wave Parameters

This is not an entry itself, but a subsection in which we will include the needed parameters to correctly define the waves for each of the wave theories.

(R)

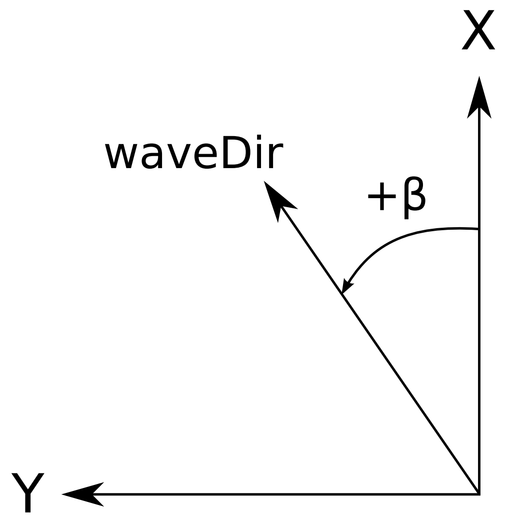


Figure 4.1: Angular reference system.

- **waveHeight** in metres
- **wavePeriod** in seconds
- **wavePhase** in radians
- **waveDir** in degrees

waveHeight and **wavePeriod** are necessary, but if **wavePhase** is not given, it is taken as $3\pi/2$ by default. The same happens to **waveDir**, which is the direction of propagation of the wave, measured in degrees, with 0°being the X axis and 90°the Y one (see figure 4.1); by default it takes value 0.

These configuration parameters are applicable to all the theories, but the **stream-function** needs additional information as given by Fenton program, as follows:

- **waveLength** in metres
- **uMean** in m/s (mean fluid speed in the frame of the wave)
- **Bj** (nondimensional)
- **Ej** (nondimensional)

Note that the results provided in **Solution.txt** are all nondimensional, so that to obtain the wave length in metres, the value obtained has to be multiplied by the water depth (h). Similarly, to obtain the mean fluid speed in the frame of the wave, the given value has to be multiplied by $\sqrt{gh}$. **Bj** and **Ej** are used as provided.

In the second time step other magnitudes will appear within the boundary condition variables, namely: **waveLength**, which is calculated according to the given theory, and

waterDepth, which is automatically calculated at the first time step. **waterDepth** can also be set to any value (greater than 0), so the given value will be used. Take care, as it is used to calculate the correction velocity for the absorption.

(I)

- **waveHeights** in metres
- **wavePeriods** in seconds
- **wavePhases** in radians
- **waveDirs** in degrees
- **secondOrder** (bool)

Note that the names are the same as before but ending with an "s", as they are lists of numbers (scalarList type). This time all of them are necessary. Additionally **secondOrder** (bool) can be set to activate second order wave generation.

(S)

The solitary wave can be seen as a limit case for a regular type, only needing:

- **waveHeight** in metres
- **waveDir** in degrees

(W)

The piston-type wavemaker velocity profile also needs special series, 2 or 3, depending on the **waveType**, with one of them always being **timeSeries**:

- **timeSeries** in seconds
- **paddlePosition** in metres
- **paddleVelocity** in m/s
- **paddleEta** in metres (free surface at the paddle)

An important fact is that **timeSeries** does not have to be provided with a homogeneous sampling rate, as the values are linearly interpolated between the given times.

Additionally, another variable called **tuningFactor** can be introduced for this wave type. It is a factor by which the velocities and free surface elevation are multiplied, and can be used as a tuning factor to match the expected laboratory results. The reason for including it is that as the numerical boundary does not move, a calibration factor may be needed. By default it takes the value 1.

(R||I||S||W)

Applicable to all of them is **nPaddles**, the (integer) number of vertical slices in which the patch is divided. This parameter plays an important role in the absorption, and also in the directional wave generation. By default it takes the value 1, i.e. the whole patch will act as an individual element.

As each of the paddles is defined by its centroid coordinates, an adequate number of paddles is needed to correctly reproduce the directionality, and also to absorb at the same time in a 3D manner using the 2D theory. As a rule of thumb, having 5 vertical columns of cells per paddle works well.

Please do not push **nPaddles** further than the number of vertical columns of cells, as this may lead to errors due to machine precision and round off errors.

Sometimes the initial conditions have to be tapered in order to obtain a more smooth response on the boundary. This applies especially for irregular sea states, or to directional waves, which at the starting point of the simulation can have a number of wave crests and troughs on the generation patch. In order to solve this issue, **tSmooth** (smoothing time, in seconds) is introduced as a fade in time in which both the free surface elevation and the velocities are multiplied by a factor which linearly varies from 0 at t = 0 s to 1 at t = tSmooth. By default **tSmooth** takes the value -1, i.e. it is not used.

Wave Absorption

Wave absorption use is controlled by a boolean variable: **genAbs**. If it is set to **true** you have wave generation plus active wave absorption, otherwise it is disconnected. By default it is set to **false**, although we recommend having it always connected to prevent the increase of the mean water level due to the unbalanced water inflow-outflow between the wave crest and trough.

nPaddles also controls how many independent absorption zones exist, allowing for a better 3D absorption when it is larger than 1. The same rules apply as seen before.

Quasi-3D absorption can be switched on by setting the direction in which the correction velocity will be applied with **absDir**. It has to be given in degrees, following the same methodology as in **waveDir**. The direction has to point to the inside of the domain, otherwise it will be useless and counterproductive. If a number higher than 360 is given, the boundary condition chooses the direction perpendicular to the paddle. This is the case by default.

All the parameters needed for wave generation are summarized in figure 4.2. The words in the square boxes are the parameters, and the ones in the rounded boxes are the options. The variables inside the circles (for the wavemaker type) indicate that the parameter next to them is needed if the **waveTheory** includes such a string.

4.3 Pure Active Wave Absorption Settings

Active wave absorption settings do not need an external dictionary file, as they are controlled within the boundary condition (i.e. within **U** file).

The necessary variables depend on the type of absorption that you are using; we will start with the full 3D theory, as it requires less parameters:

nPaddles, the number of total individual elements for absorption, as has already been explained. **nEdgeMin** and **nEdgeMax**, the number of paddles in the min/max extreme which are capable of extracting water only (for stability reasons, as presented in chapter 3). Min extreme is considered the one with minimum X coordinate if $\Delta X \geq \Delta Y$ of the patch, and the one with minimum Y coordinate otherwise. If not defined, these latest parameters take the value 0, so that all the paddles behave normally.

The 2D absorption theory (even applied to a 3D domain) can be used with the very same parameters. If Quasi-3D needs to be activated an additional parameter must be defined: **absorptionDir**, the angle at which the correction velocity is applied, again following figure 4.1, and pointing inwards the domain.

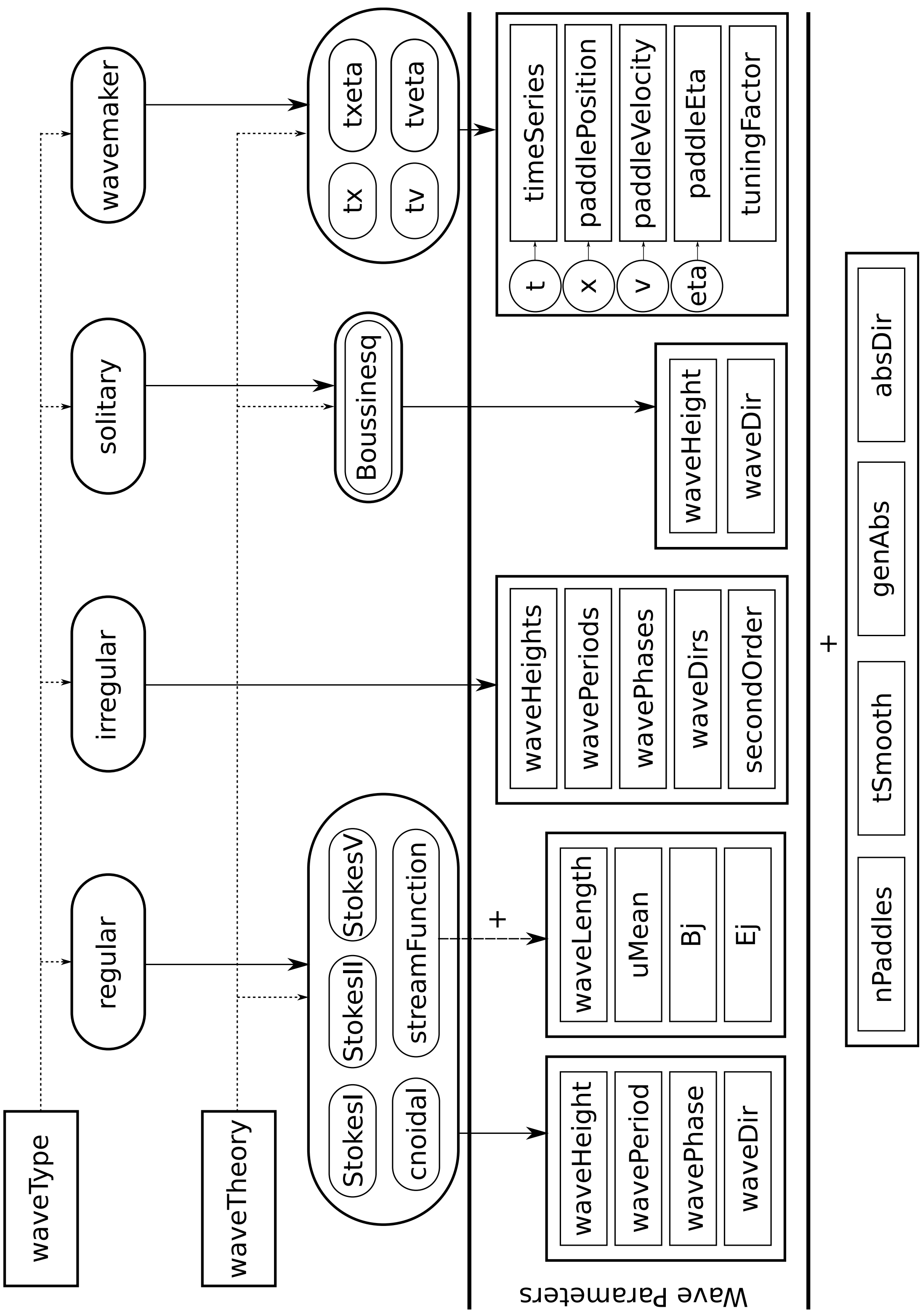


Figure 4.2: Wave generation and absorption parameters.

Appendix A
Reference

This appendix is intended to be a short guide to quickly check the most widely used features in OpenFOAM®.

A.1 Boundary conditions

A reference guide with the boundary conditions and the type to use for each variable is presented as follows. In the majority of the cases the field **value** has been set to 0. Where 0 would lead to a floating point error, a very small value has been used instead. If the case starts from a rest state, it is reasonable to use these values.

In the case of the alpha1 or p_rgh field this situation may not necessarily be true, but having these values for the initial state does not distort the calculation, as they are calculated independently at each time step. Furthermore this simple fact makes ParaView more stable at the time of loading t = 0, otherwise (if it crashes) you may have to check the **Skip Zero Time** option. The **zeroGradient** type does not need the field **value**.

alpha1 [0 0 0]

Inlet:

```
type            IH_Waves_InletAlpha;
waveDictName    IHWavesDict;
value           uniform 0;
```

Outlet & Wall:

```
type            zeroGradient;
```

Open:

```
type            inletOutlet;
inletValue      uniform 0;
value           uniform 0;
```

p_rgh [1 -1 -2]

Inlet & Outlet & Wall:

```
type            buoyantPressure;
value           uniform 0;
```

Open:

```
type               totalPressure;
p0                 uniform 0;
U                  U;
phi                phi;
rho                rho;
psi                none;
gamma              1;
value              uniform 0;
```

U [0 1 -1]

Inlet:

```
type               IH_Waves_InletVelocity;
waveDictName       IHWavesDict;
value              uniform (0 0 0);
```

Outlet:

```
type               IH_3D_2DAbsorption_InletVelocity;
absorptionDir      600.0;
nPaddles           1;
nEdgeMin           0;
nEdgeMax           0;
value              uniform (0 0 0);
```

Outlet:

```
type               IH_3D_3DAbsorption_InletVelocity;
nPaddles           10;
nEdgeMin           0;
nEdgeMax           0;
value              uniform (0 0 0);
```

Open:

```
type               pressureInletOutletVelocity;
value              uniform (0 0 0);
```

Wall:

```
type               fixedValue;
value              uniform (0 0 0);
```

k [0 2 -2]

Inlet & Outlet:

```
type               zeroGradient;
```

Open:

```
type               inletOutlet;
inletValue         uniform 0.0001;
value              uniform 0.0001;
```

Wall:

```
type               kqRWallFunction;
value              uniform 0.0001;
```

epsilon [0 2 -3]

Inlet & Outlet:

```
type                zeroGradient;
```

Open:

```
type                inletOutlet;
inletValue          uniform 0.0001;
value               uniform 0.0001;
```

Wall:

```
type                epsilonWallFunction;
value               uniform 0.0001;
```

omega [0 0 -1]

Inlet & Outlet:

```
type                zeroGradient;
```

Open:

```
type                inletOutlet;
inletValue          uniform 0.0001;
value               uniform 0.0001;
```

Wall:

```
type                omegaWallFunction;
value               uniform 0.0001;
```

nut [0 2 -1]

Inlet & Outlet:

```
type                calculated;
value               uniform 0;
```

Open:

```
type                calculated;
value               uniform 0;
```

Wall:

```
type                nutWallFunction;
value               uniform 0;
```

A.2 IHWavesDict

The usage of the wave generation and active wave absorption boundary conditions is presented in chapter 4, with all the parameters needed and the default values they take if not set. In this appendix we reference different dictionaries, one or more for each wave type (regular, irregular, solitary, wavemaker).

Regular

Cnoidal waves:

```
waveType        regular;
waveTheory      cnoidal;
genAbs          1;
absDir          0.0;
nPaddles        1;
waveHeight      0.45;
wavePeriod      3;
waveDir         0.0;
wavePhase       4.71238898;
```

Streamfunction:

```
waveType        regular;
waveTheory      streamFunction;
genAbs          1;
absDir          0.0;
nPaddles        1;
waveHeight      0.45;
wavePeriod      3.0;
waveLength      9.088;
uMean           3.102;
waveDir         0.0;
wavePhase       4.71238898;
Bj
10
(
2.82859350414499e-01
5.47232613923276e-02
9.82759610783782e-03
1.31281446426603e-03
6.95356271507966e-05
1.91627990758790e-05
4.67599434610103e-06
5.68266097588630e-07
6.71785159064360e-07
2.03467109942793e-07
);
Ej
10
(
1.90754959626264e-01
7.54514668971716e-02
2.80425743606401e-02
1.11437276888261e-02
4.79770599639068e-03
2.20447766999100e-03
1.07033607929517e-03
5.57098975071516e-04
3.34423937410174e-04
2.71836376724167e-04
);
```

Irregular

```
waveType        irregular;
genAbs          1;
absDir          0.0;
nPaddles        1;
waveHeights
3(
0.000001
0.000001
0.000002);
wavePeriods
3(
6.215700
2.207500
10.08560);
wavePhases
3(
6.031400
0.448110
4.733100);
waveDirs
3{ 0 };
```

Solitary

```
waveType        solitary;
waveTheory      Boussinesq;
genAbs          0;
absDir          0.0;
nPaddles        1;
waveHeight      0.15;
waveDir         0.0;
```

Wavemaker

```
waveType        wavemaker;
waveTheory      tveta;
genAbs          1;
absDir          0.0;
nPaddles        1;
timeSeries
2( 0 100 );
paddleVelocity
2( 2 2 );
paddleEta
2( 0 0 );
```

Bibliography

Airy, G. B. (1845). Tides and waves. In *Encyclopedia Metropolitana. Article 192*, pages 241–396.

Baldock, T. E., Swan, C., and Taylor, P. H. (1996). A laboratory study of non-linear surface waves on water. *Philosophical Transactions of the Royal Society A*, 354:649–676.

Barthel, V., Mansard, E., Sand, S., and Vis, F. (1983). Group bounded long waves in physical models. *Ocean Engineering*, 10(4):261–294.

Berberovic, E., Hinsberg, N. v., Jakirlic, S., Roisman, I., and Tropea, C. (2009). Drop impact onto a liquid layer of finite thickness: Dynamics of the cavity evolution. *Physical Review E*, 79.

Burcharth, H. and Andersen, O. (1995). On the one-dimensional steady and unsteady porous flow equations. *Coastal Engineering*, 24(3-4):233–257.

Christensen, M. and Frigaard, P. (1994). Design of absorbing wave-maker based on digital filters. *Proceeding, Waves - Physical and Numerical Modelling, Vancouver*, (100-109).

Dean, R. G. and Dalrymple, R. A. (1991). *Water wave mechanics for engineers and scientists.*, volume 2 of *Advanced Series on Ocean Engineering.* World Scientific, Singapore.

Engelund, F. (1953). On the laminar and turbulent flow of ground water through homogeneous sand. *Transactions of the Danish Academy of Technical Sciences*, 3.

Fenton, J. D. (1988). The numerical solution of steady water wave problems. *Computers & Geosciences*, 14(3):357–368.

Higuera, P., Lara, J., and Losada, I. (2013a). Realistic wave generation and active wave absorption for navier-stokes models: Application to openfoam. *Coastal Engineering*, 71:102–118.

Higuera, P., Lara, J., and Losada, I. (2013b). Simulating coastal engineering processes with openfoam. *Coastal Engineering*, 71:119–134.

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.* In print.

Jasak, H. (1996). *Error analysis and estimation for the finite volume method with applications to fluid flows.* PhD thesis, Imperial College of Science, Technology and Medicine.

Kissling, K., Springer, J., Jasak, H., Schütz, S., Urban, K., and Piesche, M. (2010). A coupled pressure based solution algorithm based on the volume-of-fluid approach for two or more immiscible fluids. In *V European Conference on Computational Fluid Dynamics, ECCOMAS CFD.*

Lara, J., Garcia, N., and Losada, I. (2006). Rans modelling applied to random wave interaction with submerged permeable structures. *Coastal Engineering*, 53:395–417.

Lara, J. L., Ruju, A., and Losada, I. J. (2011). Reynolds averaged navier-stokes modelling of long waves induced by a transient wave group on a beach. *Proceedings of the Royal Society A*, 467:1215–1242.

Le Méhauté, B. (1976). *Introduction to Hydrodynamics and Water Waves.* Springer-Verlag, New York.

Lee, J. J., Skjelbreia, E., and Raichlen, F. (1982). Measurement of velocities in solitary waves. *Journal of Waterway, Port, Coastal and Ocean Engineering*, 108:200–218.

Longuet-Higgins, M. S. and Stewart, R. (1960). Change in the form of short gravity waves on long waves and tidal currents. *Journal of Fluid Mechanics*, 8:565–583.

Losada, I., Gonzalez-Ondina, J., Diaz, G., and Gonzalez, E. (2008). Numerical simulation of transient nonlinear response of semi-enclosed water bodies: model description and experimental validation. *Coastal Engineering*, 55(1):21–34.

Mendez, F., Losada, I., and Losada, M. (2001). Wave-induced mean magnitudes in permeable submerged breakwaters. *Journal of Waterway, Port, Coastal and Ocean Engineering*, 127:7–15.

Pengzhi, L. and Liu, P.-F. (1999). Internal wave-maker for navier–stokes equations models. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 125:207–215.

Rienecker, M. M. and Fenton, J. D. (1981). A fourier approximation method for steady water waves. *Journal of Fluid Mechanics*, 104:119–137.

Rusche, H. (2002). *Computational fluid dynamics of dispersed two-phase flows at high phase fractions.* PhD thesis, Department of Mechanical Engineering, Imperial College of Science, Technology & Medicine, London.

Sand, S. E. (1982). Long waves in directional seas. *Coastal Engineering*, 6:195–208.

Schäffer, H. A. and Klopman, G. (2000). Review of multidirectional active wave absorption methods. *Journal of Waterway, Port, Coastal and Ocean Engineering*, pages 88–97.

Skjelbreia, L. and Hendrickson, J. (1960). Fifth order gravity wave theory. *Proceedings 7th Coastal Engineering Conference*, pages 184–196.

Svendsen, I. A. (2006). *Introduction to Nearshore Hydrodynamics.*, volume 24 of *Advanced Series on Ocean Engineering.* World Scientific, Singapore.

Torres-Freyermuth, A., Lara, J., and Losada, I. (2010). Numerical modelling of short- and long-wave transformation on a barred beach. *Coastal Engineering*, 57:317–330.

Troch, P. and De Rouck, J. (1999). An active wave generating-absorbing boundary condition for vof type numerical model. *Coastal Engineering*, 38(4):223–247.

Wei, G. and Kirby, J. (1995). Time-dependent numerical code for extended boussinesq equations. *Journal of Waterway, Port, Coastal and Ocean Engineering*, 121(5):251–261.

Wellens, P. (2012). *Wave simulation in truncated domains for offshore applications.* PhD thesis, Delft University of Technology.