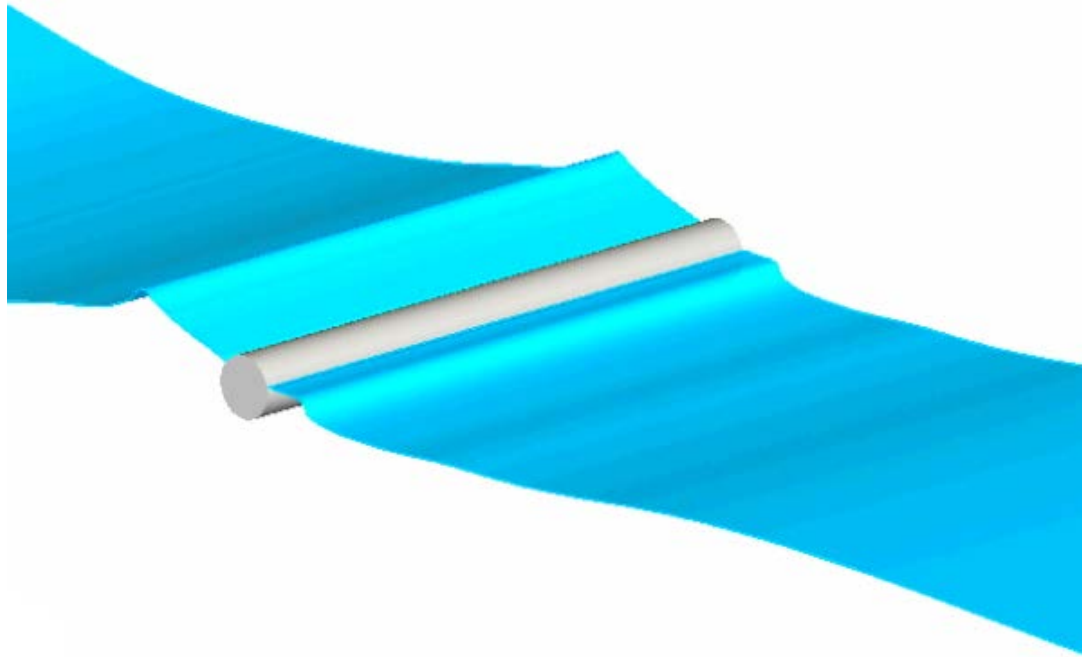


CHALMERS



Simulation of Wave Induced Forces on Semi Submerged Horizontal Cylinders Using OpenFOAM[®]

Master of Science Thesis in the Master Degree Programme Naval Architecture

JONAS ANDERSSON

Department of Shipping and Marine Technology
Division of Sustainable Ship Propulsion
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011
Master's thesis X-11/271

MASTER'S THESIS X-11/271

Simulation of Wave Induced Forces on Semi Submerged
Horizontal Cylinders Using OpenFOAM[®]

Master of Science Thesis in the Master Degree Programme Naval Architecture

JONAS ANDERSSON

Department of Shipping and Marine Technology
Division of Sustainable Ship Propulsion
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

Simulation of Wave Induced Forces on Semi Submerged Horizontal Cylinders
Using OpenFOAM®

© JONAS ANDERSSON, 2011

Master's Thesis X-11/271
Department of Shipping and Marine Technology
Division of Sustainable Ship Propulsion
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
Horizontal cylinder in waves from one of the simulations in Section 6.2.

Department of Shipping and Marine Technology
Göteborg, Sweden 2011

Simulation of Wave Induced Forces on Semi Submerged Horizontal Cylinders Using OpenFOAM®

Master of Science Thesis in the Master Degree Programme Naval Architecture

JONAS ANDERSSON

Department of Shipping and Marine Technology

Division of Sustainable Ship Propulsion

Chalmers University of Technology

ABSTRACT

The work presented in this thesis documents the simulation of wave induced forces on semi submerged horizontal cylinders in OpenFOAM, utilizing the blending method for generation and damping of waves. The blending method is not included with OpenFOAM and thus required development before it could be employed. Both a linear Airy wave model and a fifth order stokes wave model are implemented. The blending method is implemented as an extension of the laminar multiphase solver `interFoam`, utilizing the volume of fluid method for capturing the interface and the finite volume method for spatial discretization.

A validation study is carried out aiming to confirm that the simulated waves accurately represent reality. Simulation results are compared with experimental data, in this study the property compared are wave induced forces on a partially submerged cylinder. Time force histories for three different cylinder axis submergences and wave amplitudes are presented. The agreement between experimental forces and simulation forces is good.

A verification study is attempted. This study shows significant deformation of the surface profile, most notably a raising of the trough and a lowering of the crest. The cause of the deformation is still unknown.

Key words: OpenFOAM, Multiphase flow, Wave generation and damping, Wave Structure Interaction.

PREFACE

The goal of this study aims towards developing a numerical wave tank for analysis of ship hydrodynamics. The thesis is submitted as partial fulfillment of a Master of Science degree in Naval Architecture at Chalmers University of Technology, Göteborg. The work has been carried out from January to July 2011 as an internship at the Department of Research and Development, Single Buoy Moorings, Monaco, in cooperation with the Department of Shipping and Marine Technology, Chalmers University of Technology.

ACKNOWLEDGEMENTS

I would like to thank my supervisor at Single Buoy Moorings, Computational Fluid Dynamics Engineer Jean-Luc Pélerin, who generously wrote much of the code and patiently guided me for countless hours. I also wish to extend my gratitude to my examiner Dr. Claes Eskilsson at Chalmers University of Technology.

Monaco, July 2011

Jonas Andersson

Contents

1	INTRODUCTION	1
1.1	Aims and Objectives	2
1.2	Delimitations	2
2	WAVE THEORY	3
2.1	Airy Wave	4
2.2	Fifth Order Stokes Wave	5
3	COMPUTATIONAL FLUID DYNAMICS	7
3.1	Governing Equations	7
3.2	Finite Volume Method	7
3.3	Volume of Fluid Surface Capturing	7
3.4	Implicit and Explicit Numerical Schemes	8
4	SOFTWARE	9
4.1	OpenFOAM	9
4.1.1	blockMesh	10
4.2	ParaView	10
4.3	Matlab	10
5	WAVE GENERATOR DEVELOPMENT	11
5.1	Blending Method	11
5.1.1	Blending Functions	12
5.1.2	Implementation in OpenFOAM	14
5.2	Discussion Wave Generator Development	15
6	FORCES ON SEMI SUBMERGED CYLINDERS	19
6.1	Previous Research on the Topic	19
6.1.1	Dixon et al.'s Experimental Study	19
6.1.2	Westphalen et al.'s CFD Solver Validation	21
6.2	OpenFOAM Numerical Wave Tank Validation	21
6.2.1	Motivation of Chosen Case Setups	21
6.2.2	Wave Parameters	22
6.2.3	Basin Layout	24
6.2.4	Mesh Generation	25
6.2.5	Force Measurement	27
6.3	Results and Discussion	28
7	CONCLUSIONS	35
8	REFERENCES	37

1 Introduction

Within ship hydrodynamics a number of coefficients are required in order to predict how ships behave. For instance a sea keeping study might require the roll damping coefficient, while a load analysis using the Morison equation would require two coefficients, inertia and drag. These coefficients are usually determined using small scale model experiments in wave tanks.

Physical experiments involve a number of disadvantages compared to numerical simulations. Manufacturing of the model is time consuming, potentially causing costly delays when several iterations of the design are required. The model size is limited by the size of the wave tank, which generally is several orders of magnitude smaller than a full size ship. This causes a problem because scaling of the results from model scale to full scale inevitably involves a loss of accuracy. This all adds up and makes wave tank experiments a costly venture.

Computational Fluid Dynamics (CFD) can be used to create a numerical wave tank, consequently providing an alternative method for determining the sought coefficients without the previously mentioned disadvantages. However, CFD is not yet capable of replacing physical experiments. A CFD solution can never be more accurate than the underlying physical models. Currently CFD should be considered a complement to real experiments.

Single Buoy Moorings, from now on referred to as SBM, designs Floating Production, Storage and Offloading (FPSO) platforms, mooring terminals and other offshore equipment. A firm understanding of how their equipment behaves in waves is thus of central importance for SBM.

1.1 Aims and Objectives

The purpose of this project is to investigate the usage of the open source CFD toolbox OpenFOAM¹, for creation of a numerical wave tank, and its possible usefulness within SBMs industrial projects.

The direction of the study is aimed towards a numerical wave tank for determination of ship hydrodynamic coefficients. With an understanding that this is a very ambitious objective, considering the time and resources dedicated to the project, the objective is rather to investigate the required building blocks and to provide SBMs personnel with a better understanding of the subject.

A rough outline of the building blocks required for a numerical wave tank for the determination of ship hydrodynamic coefficients would include: a CFD solver for multiphase flows, an interface tracking method, a technique for generating and absorbing waves, wave models, couplings between wave induced forces and body motions and capabilities to handle moving objects.

1.2 Delimitations

A prerequisite for this study is that OpenFOAM must be used. There are many alternatives available, however most of them come with license constraints, limiting the possibilities to modify and extend the code, see Section 4.1. SBM specifically wish to investigate the current usefulness of OpenFOAM for wave generation purposes. Since some development had previously been performed using OpenFOAM 1.5dev [1], this version shall be used also for further development.

The technique used for wave generation and absorption is the relaxation- or blending method. This is a method not initially included in the OpenFOAM toolbox, thus requiring implementation.

¹This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM[®] and OpenCFD[®] trade marks.

2 Wave Theory

There are several approaches to generating waves in a numerical wave tank. The same approach as in an experimental wave tank could be used, i.e. a moving paddle. This method would however require knowledge of which paddle motion corresponding to the desired wave shape, as well as a method for moving the paddle, introducing an unwanted complication. Another approach, used in this project, is generating an analytical wave model.

No single wave theory describes all types of waves. Currents, water depth and wave steepness all significantly affect the wave shape. Thus knowledge about these parameters are required in order to choose a suitable wave model. An improper choice of wave model will result in a wave unable to retain the generated shape.

The depth of the numerical wave tank in this project is always chosen to be deep, meaning the water depth is always larger than half the wavelength, $h > \frac{1}{2}\lambda$. No current is applied and the steepness is varied. The two wave models implemented in this project are the linear Airy wave and Fentons nonlinear fifth order Stokes wave [2]. The difference between the two wave models surface profiles can be seen in Figure 1.

2.1 Airy Wave

The so called Airy wave theory is a linear, or first order, wave model. The surface profile is symmetric about the mean water level so that peaks have the same shape as troughs, see Figure 1. It is only valid for small amplitude waves, meaning that the wave amplitude has to be much smaller than the wave length. It is based on potential flow and thus does not account for viscosity or vorticity. Since it is linear it is possible to superpose solutions for different wave frequencies and propagation directions and thus achieve an irregular sea state [3].

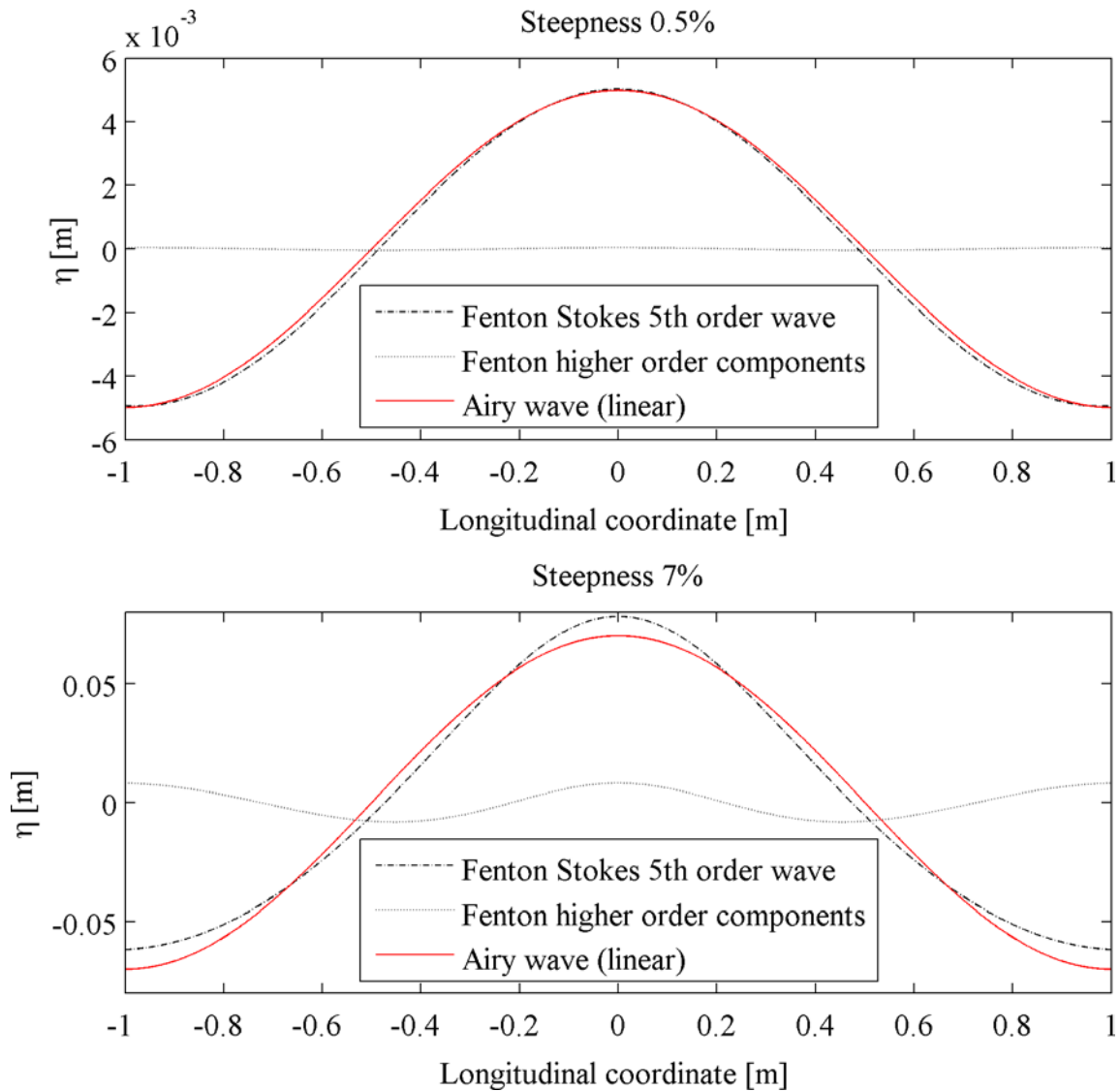


Figure 1 Comparison of surface profiles for Airy and Fenton Stokes 5th order wave models.

The Airy wave model can be computed according to equations (2.1)-(2.7):

$$\omega = 2\pi f \quad (2.1)$$

$$k = \frac{\omega^2}{g} \quad (2.2)$$

$$\lambda = \frac{2\pi}{k} \quad (2.3)$$

$$\eta = a \cdot \cos(kx - \omega t) \quad (2.4)$$

$$u = \frac{a\omega \cosh(k(h+z)) \cos(kx - \omega t)}{\sinh(kh)} \quad (2.5)$$

$$w = \frac{a\omega \sinh(k(h+z)) \sin(kx - \omega t)}{\sinh(kh)} \quad (2.6)$$

$$p = \rho g \eta \frac{\cosh(k(h+z))}{\cosh(kh)} \quad (2.7)$$

Where ω is the angular frequency, f the frequency, k the angular wave number, g gravity, λ the wavelength, η the surface elevation, a wave amplitude, u the horizontal particle velocity, w the vertical particle velocity, h the still water depth and p the pressure.

2.2 Fifth Order Stokes Wave

Fentons fifth order stokes wave [2, 4] is uniquely defined by three physical dimensions, the mean water depth d , the wave height H , and the wavelength λ . Several other wave theories employ the wave period instead of wavelength. However in most real applications there exists a current, and the apparent wave period is actually Doppler-shifted. When using a Doppler-shifted wave period without knowledge about the current, the problem is under specified and thus does not uniquely define the wave. No current velocity is applied in this study, however possible future applications for this code likely do.

As seen in equation (2.8), the free surface profile η is much more complex for a fifth order wave in comparison to the Airy wave, see Figure 1. The higher order wave has higher and narrower crests and flatter shallower troughs compared to the Airy wave.

$$\begin{aligned} k\eta(x) = & kd + \epsilon \cos(kx) + \epsilon^2 B_{22} \cos(2kx) \\ & + \epsilon^3 B_{31} (\cos(kx) - \cos(3kx)) \\ & + \epsilon^4 (B_{42} \cos(2kx) + B_{44} \cos(4kx)) \\ & + \epsilon^5 (-(B_{53} + B_{55})) \cos(kx) \\ & + B_{53} \cos(3kx) + B_{55} \cos(5kx) \end{aligned} \quad (2.8)$$

In equation (2.8) $\epsilon = kH/2$ is the dimensionless wave height and B_{ij} are dimensionless coefficients. Fentons fifth order Stokes model includes 25 of these coefficients, each described by an intricate formula. These formulas can be found in [2] and [4]. Fenton lists the following equation [4] for determining the highest stable wave height as a function of wavelength and depth, equation (2.9):

$$\frac{H_m}{h} = \frac{0.141063\frac{\lambda}{h} + 0.0095721\left(\frac{\lambda}{h}\right)^2 + 0.0077829\left(\frac{\lambda}{h}\right)^3}{1 + 0.0788340\frac{\lambda}{h} + 0.0317567\left(\frac{\lambda}{h}\right)^2 + 0.0093407\left(\frac{\lambda}{h}\right)^3} \quad (2.9)$$

Where H_m is the maximum wave height, h the depth and λ the wavelength.

3 Computational Fluid Dynamics

In this project Computational Fluid Dynamics is used to solve the flow in a numerical wave tank. CFD is a branch of fluid mechanics utilizing numerical methods to solve fluid flows. This chapter introduces the governing equations behind the CFD software used in this project, OpenFOAM. The solver used is `interFoam`, a solver for two incompressible, isothermal immiscible fluids using the volume of fluid (VOF) method for capturing the interface.

3.1 Governing Equations

Navier-Stokes equations are nonlinear partial differential equations believed to exactly describe all fluid flows. The governing equations for a time-dependent three-dimensional fluid flow and heat transfer of a compressible Newtonian fluid can be written as equation (3.1) [5].

$$\frac{\partial(\rho\phi)}{\partial t} + \text{div}(\rho\mathbf{u}\phi) = \text{div}(\Gamma \text{grad } \phi) + S_\phi \quad (3.1)$$

Where ϕ is a general variable representing e.g. velocity. The first term represents the time rate of change of ϕ within a control volume and the second term represents convection. On the right hand side the first term represents diffusion, where Γ is a diffusion coefficient, and S_ϕ is the source term.

Few exact solutions exist and thus the partial differential equations need to be solved numerically for most engineering problems. Navier-Stokes equations can be solved directly, however it is too computationally prohibitive for most engineering problems. Thus simplifying assumptions and models are required in order to make it economically viable. An assumption made in this project is that the fluid is inviscid, resulting in the so called Euler equations.

3.2 Finite Volume Method

OpenFOAM uses the finite volume method for spatial discretization of the partial differential equations. The domain is divided into control volumes, referred to as cells. The differential form of the partial differential equations are integrated over each cell. Because the flux into a cell is always equal to the flux out of a cell, the finite volume method is inherently conservative. Quantities of interest, e.g. pressure or velocity, are stored at the cells centroid. Values in between the cell centroids are interpolated.

3.3 Volume of Fluid Surface Capturing

The `interFoam` solver employs the VOF method for capturing the interface. The VOF method utilizes a property called phase fraction to capture the interface. In OpenFOAM 1.5dev the phase fraction is denoted γ while newer versions denote α . The phase fraction is convectively transported by the velocity field, see the phase fraction transport equation (3.2).

$$\frac{\partial\gamma}{\partial t} + \mathbf{v} \cdot \text{grad } \gamma = 0 \quad (3.2)$$

The phase fraction has a value of zero in cells filled with air and a value of one in cells filled with water. Cells at the free surface contain both air and water and thus have a value between zero and one.

The VOF formulation in `interFoam` tends to artificially diffuse the interface between air and water. One method for keeping the surface interface sharp is artificial compression where a pressure is applied normal to the surface. In OpenFOAM the magnitude of the artificial volume force is controlled via the `cGamma` setting.

3.4 Implicit and Explicit Numerical Schemes

Since the choice of numerical schemes turned out to have a large influence on problems experienced during the project, such as surface diffusion and unrealistic air velocities, a brief description of the differences between implicit and explicit schemes are given in this section. A more thorough description can be found in Versteeg [5]. For details regarding the problems experienced, see Section 5.2.

With an explicit numerical scheme the solution for the current timestep is computed only from known previous solutions and thus can be solved directly, see equation (3.3).

$$Y(t) = f(Y(t - \Delta t)) \quad (3.3)$$

$Y(t)$ is the solution at the current timestep t , Δt is a time increment and thus $Y(t - \Delta t)$ is the solution at the previous timestep.

In an implicit numerical scheme on the other hand, the solution is evaluated also from the unknown current timestep. Since the current solution is present on both sides of the equation it cannot be solved directly, see equation (3.4).

$$Y(t) = f(Y(t - \Delta t), Y(t)) \quad (3.4)$$

Implicit schemes are much more stable than explicit schemes. Explicit schemes needs to fulfill the Courant-Friedrich-Levy (CFL) condition in order to guarantee stability, see equation (3.5).

$$\frac{U \cdot \Delta t}{\Delta x} = CFL \quad (3.5)$$

U is the velocity, Δt is the timestep length and Δx is the cell length. For this condition to be fulfilled the distance travelled during one timestep is required to be less than one cell length. Hence high velocities and small cell sizes requires very small timesteps.

Time marching is different for implicit and explicit schemes. With implicit time marching it is possible to maintain stability while advancing larger temporal distances than dictated by the CFL condition. This is an attractive behavior for steady state simulations. The converged solution is correct, however the iteration steps between initialization and converged solution is not time accurate when using implicit time marching.

4 Software

This chapter contains brief introductions to the software used in this project. All work is performed under Linux, which with its powerful scripting capabilities lends itself well for this particular type of workflow. Several alternatives to the specified softwares were tried and considered. However, due to the repetitive tasks this type of simulation work often involve, software alternatives which provide good scripting and automation capabilities ended up being the preferred choice.

4.1 OpenFOAM

OpenFOAM, short for Open Field Operation And Manipulation, is a free open source CFD toolbox [6]. It is a library of C++ code, available for anyone to customize and extend. OpenFOAM is written using an object oriented approach with a syntax closely resembling the differential equations being solved. This yields a very modular software design which eases the transition from physical model to code, making it easy to modify and extend [7]. In this study OpenFOAM 1.5dev has been used [1].

OpenFOAM uses the finite volume method on unstructured meshes. It is a very capable CFD tool including free-surface flows and mesh motion, among many other features. More than 80 solvers for simulation of specific problems and over 170 utility applications for meshing, post-processing and similar tasks are included [8].

Unlike most other CFD software, OpenFOAM is not controlled through a graphical user interface (GUI). Instead settings are adjusted via text files called dictionaries and everything is controlled via the command line. The lack of a GUI makes the learning curve steeper for new users, however once grasped it is a very flexible and easily automatable approach. Some documentation and support are available for OpenFOAM, but compared to commercial CFD software alternatives it is lacking.

OpenFOAM is released under the license GNU GPL 2 [9], meaning that it is free for anyone to download and use. Commercial software licenses generally limit the number of processors per simulation or the number of parallel simulations. OpenFOAM offers the flexibility of utilizing all available processors for a single simulation, or for launching a large amount of parallel simulations. Thus OpenFOAM allows utilizing the available hardware to its full potential.

The nonexistent license fee is however not the only benefit. The lack of license constraints means that OpenFOAM can be customized to suit any workflow. Tasks ordinarily requiring manual interaction can be automated. Everything from mesh generation, simulation, post-processing and plotting can be automatically performed for complex matrices of initial conditions, boundary conditions, fluid properties and geometries. All parts of an OpenFOAM simulation, from pre processing to post processing, can be automated using the python library PyFoam [10].

4.1.1 blockMesh

One of the utilities distributed with OpenFOAM is `blockMesh`, a mesh generator for simpler geometries. As with all OpenFOAM utilities, `blockMesh` is controlled through a dictionary, which facilitates easy creation of parametric meshes. A few input parameters can thus generate a complete mesh, while a few alterations to these parameters might generate a very different mesh. This makes `blockMesh` particularly well suited for studies involving families of geometrically similar meshes, such as grid dependence or optimization studies.

The preprocessing language GNU `m4` [11] is commonly utilized for parameterization of `blockMesh` dictionaries. More advanced calculations can be achieved using shell scripting, `perl` or `python`.

4.2 ParaView

ParaView is an open source post processing application for data analysis and visualization. Using the Message Passing Interface (MPI), ParaView can run in parallel, enabling fast rendering of large data sets. Post processing can be carried out interactively in 3D as for most post processing software. ParaView can however also be run in batch mode, enabling rapid and effortless repetition of identical processing tasks, and scriptable, automatic post processing [12].

All actions performed through the GUI of ParaView can be recorded as macros for quick re-use of common processing sequences. The macro feature records the GUIs actions using `python`. These `python` scripts can also be executed in batch mode, and thus fits well into the scriptable workflow of OpenFOAM.

OpenFOAM is well integrated with ParaView through the wrapper script `paraFoam`. Simulations can be visualized in ParaView without the need for converting or even recomposing parallel cases.

4.3 Matlab

Matlab has a wide range of applications within scientific computing, such as algorithm development, image processing and data visualization. A large number of add-on toolboxes can further increase the feature set. The programming syntax in Matlab is easier than traditional scientific programming languages such as `C/C++` or `Fortran` [13].

Data presentation in Matlab allows for highly detailed control over the visualized results, making it the authors' software of choice for generating publication grade figures. Matlab scripts can be run interactively, but also in batch mode from the command line and thus integrates well with the other software used through scripting.

5 Wave Generator Development

Working with waves travelling across the limited spatial domain of a numerical wave tank first requires the waves to be generated. The length of a numerical wave tank is generally smaller than the distance travelled by a wave during a normal simulation duration. Thus the wave is likely to reach the end of the domain and reflect off of it. Reflected waves are generally an unwanted phenomenon since they interfere with the tests being conducted in the numerical wave tank. In order to prevent waves reflecting off of the domain boundaries wave absorption techniques can be employed.

Several techniques are available for wave generation in numerical wave tanks. Moving paddles, similar to those used for wave generation in real wave tanks, can be recreated and analytical formulas can connect each desired wave type to the corresponding paddle motion required. This method requires the challenging task of simulating a moving paddle. Other methods for generating waves make use of the possibility to control mass sources within the continuity equation and thereby recreating the mass flux of the desired wave type [14].

Absorption of an incoming wave is an even more challenging task than generation of the wave. In real wave tanks sponge layers are often used to absorb incoming waves and prevent them from reflecting. A common approach for numerical wave tanks is to implement a Sommerfeld boundary condition, where a first order wave is prescribed at the boundary. The prescribed wave needs to match the incoming wave exactly regarding both time and frequency for successful absorption. Any discrepancies between the two will result in reflections. Thus the Sommerfeld boundary condition is incapable of fully absorbing waves deformed by e.g. a rolling ship in the wave tank. Another method for damping waves is to add a viscous damping term to the mass and momentum equations [15].

5.1 Blending Method

The blending method can be utilized for both wave generation and wave absorption. Unlike wave generating methods based on periodic boundary conditions, the blending method can use simple boundary conditions, and it is capable of dissipating an incoming wave without prior knowledge on the properties of the wave, or at what exact time it will arrive.

The blending method generates waves in the interior of the basin. It functions by substituting the computed solution with an alternative solution after each time increment. The solution being enforced could for instance be a wave model or a still surface. This substitution should not be performed all at once, as it would result in large gradients and produce reflections. Instead the solutions are gradually substituted throughout the length of the basin.

An advantage with the blending method is that it can effectively dissipate incoming waves, not only in the damping end of the basin, but also at the generating end. Since the same method is used for both generating and damping the waves, it is possible to simulate very complex cases having waves propagating in several directions simultaneously, impacting the damping zones at any angle. However the blending method operate inside the actual basin and require a basin length the magnitude of several wavelengths. This significantly increases the required simulation time.

The blending method is not included with OpenFOAM and thus requires implementation. The method has been successfully implemented in OpenFOAM previously by Afshar [16] and Jacobsen [17].

Another common name for this method is the relaxation method. This term is not chosen by the author, since within CFD the term relaxation is commonly associated with a mixing of the calculated solution with zero or with a solution from a previous timestep, thereby relaxing the advance of the simulation, making it more stable.

The chosen term blending on the other hand suggests that the solution is mixed with an alternative solution, not necessarily zero or a solution from a previous timestep. This is deemed a more suitable name for this method since its purpose is not to stabilize the simulation, but rather to propagate a wave.

The blended solution is calculated according to equation (5.1),

$$\phi_b(x) = \alpha(x)\phi_c(x) + (1 - \alpha(x))\phi_a(x) \quad (5.1)$$

where ϕ_b is the new blended solution, ϕ_c is the old solution, ϕ_a is the analytical solution being generated, and $\alpha(x) \in [0,1]$ is the blending function. The first term on the right hand side, $\alpha(x)\phi_a(x)$, represents the absorption while the second term, $(1 - \alpha(x))\phi_c(x)$, controls generation.

5.1.1 Blending Functions

Generation and damping of waves using the blending method needs to be performed smoothly in order to prevent spurious wave modes and reflections. In order to achieve a smooth blending the shape of the blending functions and the length of the blending zones needs to be considered.

Engsig-Karup [15] derives blending function shapes ensuring smooth transition across zone boundaries. At the zone boundaries the solution as well as its derivatives needs to remain unchanged before and after the blending. Thus $\alpha(x)$ needs to be equal to one when bounding towards the wave tank zone, and equal to zero at walls. Homogenous $\alpha(x)$ derivatives are required near all boundaries, with the exception of boundaries where waves travel in only one direction, such as from the wave tank into the damping zone. The blending function should ideally contain no discontinuities in order to avoid reflections. Due to the interface jump between cells in a discretized mesh there will always be some discontinuities. Thus in order to keep errors small a fine mesh is required.

The blending functions used in this project are those proposed by Engsig-Karup, see equations (5.2) and (5.3) and the corresponding profiles in a zone layout in Figure 2.

$$\alpha(x) = -2x^3 + 3x^2 \quad (5.2)$$

$$\alpha(x) = 1 - x^6 \quad (5.3)$$

Equation (5.3) fulfills the requirements of vanishing derivatives at the boundaries, while equation (5.2) fulfills only the requirement at one end and is thus better suited for damping zones. The profiles can be reversed by a coordinate change, e.g. $\alpha(1 - x)$. The exponent in equation (5.3) is set to 6, in accordance with Afshar [16].

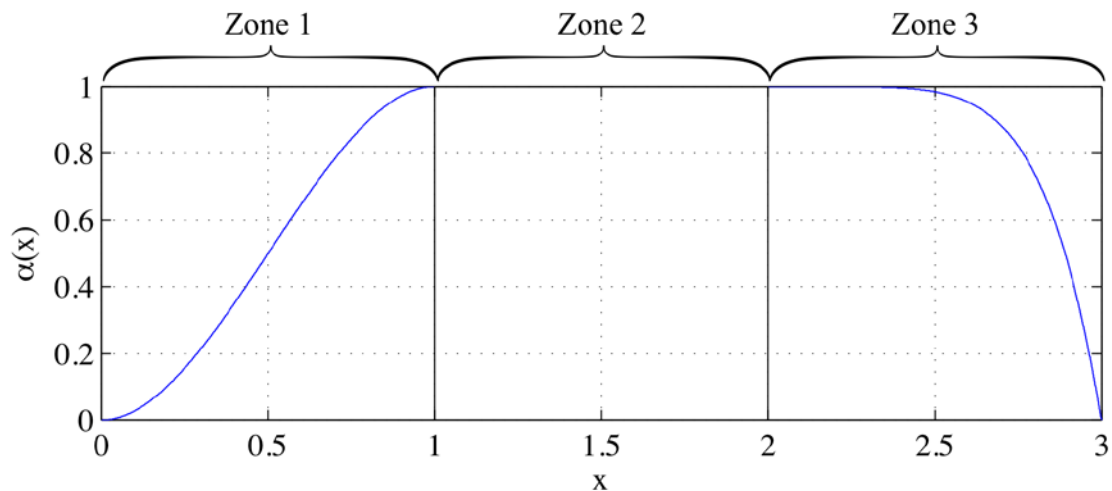


Figure 2 Zone layout including damping function profiles.

In Figure 2 $\alpha(x)$ is the damping coefficient and x is the distance from the generating zone border in the propagation direction. Zone 1 depicts the blending function profile of equation (5.2), zone 2 is the wave tank and thus no blending is performed while zone 3 contains the blending function profile of equation (5.3).

The actual length of each zone might be one to two wavelengths for the generator zone 1 while the wave tank zone 2 needs to be large enough to contain whatever is being investigated and the damping zone 3 might require one to three wavelengths depending on the incoming waves.

Zone 1 begins with a completely enforced wave model at the left wall, over which the solver has no influence. Gradually along zone 1 the enforcing of the wave model is ramped down according to equation (5.2) and the solver is given control over the solution. At the end of zone 1 there is no blending, only the solution computed by the solver which at this point propagates the waves.

This unenforced propagation continues throughout all of zone 2 which represents the actual wave tank. When the waves reach the end of zone 2 they may be very different from when they entered zone 2. Since no blending is performed in the wave tank they may have deformed due to interaction with objects in the basin.

At the beginning of zone 3 the solution computed by the solver, i.e. incoming waves, are gradually absorbed according to equation (5.3) and replaced by an enforced still surface. At the end of zone 3 the blending function derivatives are non-homogenous. This should however not matter since no wave energy should be left to reflect off of the bounding wall.

In some cases it might be necessary to use different zone layouts. Afshar [16] and Engsig-Karup [15] for instance employ two zones for generating waves. The additional zone is placed next to the basin wall in order to remove the violent behavior otherwise arising when enforcing a wave model close to the wall. A large number of zone configurations were investigated during the course of this project, and the single zone generator layout was found to yield the shortest simulation times due to its smaller size. A fluctuating pressure with high magnitude does arise at the wall, however it seems to cause no effect on the wave tank.

5.1.2 Implementation in OpenFOAM

It is possible to generate waves in OpenFOAM using the included `groovyBC` package. The blending method however is not included with OpenFOAM, and thus requires implementation.

The blending method is implemented as an extension of the solver `interFoam`, a laminar multiphase solver for incompressible fluids. The `interFoam` solver employs the volume of fluid method for capturing the interface, see Section 3.3, and spatial discretization is performed according to the finite volume method, see Section 3.2. The VOF phase fraction is computed explicitly while velocity and pressure is handled implicitly, see Section 3.4.

All settings within OpenFOAM are controlled by means of text files called dictionaries, as previously mentioned in Section 4.1. Each case consists of multiple dictionaries organized in a specific file structure, as specified in the OpenFOAM user guide [6]. In accordance with this structure, all input parameters required to generate waves using the blending method is divided into two dictionaries placed in the `constant` directory. The `waveGeneratorsDict` specifies all parameters concerning the blending zones, and `wavesDict` contains the wave parameters.

Multiple zones can be specified in the `waveGeneratorsDict` dictionary, each zone in a separate sub-dictionary. Required input parameters are the zone position, dimension and orientation, as well as the blending function and wave type to blend with. Any of the wave types specified in the `wavesDict` dictionary can be used. Some of the blending functions accept input parameters for adjusting the profile shape. For instance the exponent in equation (5.3) can easily be adjusted. All parameters can be specified for the three fields velocity, pressure and VOF phase fraction and blending can be turned on or off individually for each field. By enabling the `debug` switch a number of additional fields are saved for each timestep. The fields show the position, orientation and wave shape enforced for each zone. Each additional field require a large amount of disk space and thus the `debug` switch should only be enabled to initially in order to verify the zone layout. A sample of how a blending zone is specified in the `waveGeneratorsDict` dictionary is seen in Figure 3.

The `wavesDict` dictionary can contain parameters for several different wave types, each in a separate sub-dictionary. Each wave type sub-dictionary contains a wave model and its corresponding phase shift, propagation direction, amplitude, frequency and basin depth. The wave models implemented in this project are Fentons fifth order Stokes model [2], the first order linear Airy wave, and a still surface. A sample of how a wave is specified in the `wavesDict` dictionary is seen in Figure 4

As explained in Section 2.2 the relation between frequency and wave length is not trivial for fifth order Stokes waves. Some cases require a specific wave frequency while others require a specific wave length. The utility `infoWaves` can be used to quickly calculate the resulting wave length for any of the wave types specified in the `wavesDict` dictionary. As explained in Section 2.2, the fifth order Stokes wave model implemented requires 25 coefficients to be calculated. Fenton [2] publishes values for these coefficients for three cases. The `infoWaves` utility computes the coefficient values for the same three cases, thus providing a method for verifying that the coefficients have been accurately implemented.

The utility `initWaves` functions similarly to the OpenFOAM bundled `setFields` utility, used to initiate the VOF phase fraction field. However `initWaves` is also capable of initializing wave fields for any of the wave types specified in the `wavesDict` dictionary. Thus it is possible to initialize a simulation with a fully developed wave field over the entire basin, or a still surface.

The algorithm for the developed wave generator is thus: read zone layout and wave parameters from the `waveGeneratorsDict` and `wavesDict` dictionaries. Initialize the desired wave field using the `initWaves` utility. Start the simulation and between each time increment, blend the computed solution in the blending zones with the desired wave types.

```

blendingZone1
{
  velocityBlendingFunction {
    type      EngsigKarup1;
    exponent  6;
  }
  pressureBlendingFunction {
    type      EngsigKarup1;
    exponent  6;
  }
  gammaBlendingFunction {
    type      EngsigKarup1;
    exponent  6;
  }
  blendGamma      on;
  blendVelocity   on;
  blendPressure   off;
  reference       (-3 0 0);
  forward         (1 0 0);
  up              (0 0 1);
  width           1;
  length          1.5;
  wave            fentonFifth;
  debug          off;
};

```

Figure 3 A sample of how the blending zone parameters are specified in the `waveGeneratorsDict` dictionary.

```

fentonFifth
{
  type      Stokes5thOrder;
  g         (0 0 -9.81);
  phasesOrigin (1.15 0 -0.0045);
  direction (1 0 0);
  amplitude 0.05;
  frequency 1.04235;
  depth     0.9;
  current   0;
};

```

Figure 4 A sample of how a wave is specified in the `wavesDict` dictionary.

5.2 Discussion Wave Generator Development

During the development and subsequent utilization of the blending method wave generator a number of complications were encountered. This section documents many

of these complications in an effort to aid others working with blending method wave generation in OpenFOAM. Highlighting some of the more subtle difficulties to watch out for, detailing solutions for certain problems and offering possible resolutions for others.

The phase fraction should be closely monitored throughout the simulations to make sure that the surface is not rising or falling. The process may be quite slow and thus hard to detect, but it can have a large influence on the results. There are several possible causes for this problem. It is often related to boundary conditions or the depth of the enforced wave models not matching the actual still water depth.

The VOF surface capturing method has a tendency to diffuse the interface between air and water. As described in Section 3.3 an artificial volume force is applied to compress the surface, thereby retaining a sharp interface [18]. The amount of compression is controlled via the `cGamma` setting, or `cAlpha` in newer versions of OpenFOAM. For longer simulations some level of compression was required to prevent excessive diffusion. Afshar [16] reported that high levels of compression lead to deformation of the wave profile for longer simulations, an issue not encountered during this study. Possibly thanks to the semi-implicit solver, or none of the attempted simulations were long enough for the problems to appear.

A particularly troublesome problem is the unwanted high air velocities close to the water surface. These unnaturally high air velocities may deform the wave profile and lead to premature wave breaking. They result in higher CFL numbers and thus smaller time steps and slower simulations. The same phenomenon is reported also by Paterson [10] and Afshar [16] who suggest ignoring the air side convective term or relaxation of air velocities. The method which best solved the problems of high air velocities in this study was treating the VOF phase fraction explicitly and velocity and pressure implicitly. Initially all fields were computed explicitly, leading to high air velocities. Treating all fields implicitly did not work well together with the volume of fluid method, as the surface was quickly diffused despite high `cGamma` values. The final semi implicit solver yielded the best results.

A problem as of yet unresolved is the raising of the wave trough and decline of the wave crest encountered during the verification study. The generated waves are highly accurate inside the generating zone, but as soon as they enter the wave basin the trough quickly raises, soon followed by a decline of the crest. No combination of settings were able to solve this, suggesting that the root of the problem might lie deeper, e.g. in the semi implicit solver.

As explained in Section 5.1.1 a zone layout incorporating a generating zone flush against the wall, with full enforcement of the wave model next to the wall, is used for the simulations presented in this report. In combination with the no slip impermeable boundary condition used this gives rise to a rather violent pressure fluctuation at the wall. This has the advantage of reducing the size of the wave generator and thus reducing the required simulation time. No influence from the high magnitude pressure fluctuation was detected in the actual wave basin. The only drawback encountered as a result of this layout is that the high magnitude of the pressure fluctuation complicates investigation of the pressure field during post processing, a small inconvenience at most. A possible approach to neutralize the fluctuating high magnitude pressure field, without extending the dimensions of the wave generator, could be to implement a correspondingly fluctuating boundary condition.

This implementation of the blending method for wave generation and dissipation is currently significantly slower than other alternatives. However no systematic approach to increase its speed has been attempted during this project, leaving much room for optimization. The large zones required for generation and damping when using the blending method could possibly be reduced by finding the optimal blending function.

In the current implementation of the blending method, the aim is to dampen incoming waves before they reach the wall and thus reflect off of it. Reflecting waves encounter decreasing damping when leaving the damping zone. Perhaps it could be possible to separate incoming waves from waves travelling in the opposite direction, and to enforce different levels of damping depending on the direction of propagation. As a result the required zone sizes could effectively be reduced to half of their current size.

As explained in Section 5.1, the blending method functions by blending the computed solution with a wave model after each time increment. The speed of the blending depends on the length of the time increments. Thus zone lengths and blending functions suitable for one simulation may not suit a simulation with different timestep lengths. When enabling OpenFOAMs `adjustableRunTime` feature the time step length is automatically adjusted and thus varying throughout the simulation. Consequently the blending speed will also vary throughout the simulation. A method for making the blending speed independent of time increment length should be implemented. Perhaps by only blending at predetermined equidistant time increments.

An aspect further decreasing the speed of implementation of the blending method is that it requires a very fine mesh in the surface region to propagate a sharp wave profile. Possibly an improved version of the VOF surface capturing method could be implemented, e.g. using a local height function. In the current implementation all cells with cell center lower than the analytical wave profile get a phase fraction value of one, thus representing water, while all other cells are initialized as air. As suggested by Afshar [16], by not only initializing phase fractions of one or zero, but also values in between, a more accurate representation of the wave profile could be initialized using a coarser mesh [17].

6 Forces on Semi Submerged Cylinders

With `interFoam` extended to include wave generating and damping capabilities, see Chapter 5, and a fifth order Stokes wave model implemented see Section 2.2, a suitable validation case is sought.

The case chosen is a horizontal cylinder, fixed in the surface region. It is chosen because it has a comparatively simple setup, and the large amount of previous research on the topic offers plenty of data to validate against.

The purpose of this case is to validate the wave induced forces on a cylinder in the surface region. The assumption is that if the forces measured in the simulation match those of physical experiments, the underlying models accurately recreates reality.

6.1 Previous Research on the Topic

Since the purpose of this case is to validate simulation results against experimental data, a good understanding of the experimental setup is required to understand the motivations behind the simulation setup. In order to facilitate a better understanding of the experimental data, a summary of previous research on the topic is presented in this section. The research performed by Dixon et al. [19] and Westphalen et al. [20] are described in particular detail in Sections 6.1.1 and 6.1.2, since they contain the data chosen to be validated against.

The Keulegan-Carpenter number N_{KC} is the ratio between drag forces and inertial forces. A low N_{KC} thus indicates that the loads are dominated by inertial forces with a negligible contribution from drag forces. For linear waves it can be calculated according to equation (6.1).

$$N_{KC} = \frac{2\pi a}{D} \quad (6.1)$$

Wave induced forces on fixed horizontal cylinders is a topic with a large amount of published research. Prasad [21] reviewed a large number of reports on the topic. Easson et al. [22] described the force spectra in Gaussian seas while Kaplan and Silbert [23] investigated slamming. More recently, Chaplin [24, 25], Westphalen et al. [20] and Hu et al. [26] studied the forces on fixed horizontal cylinders in the surface region. Westphalen and Hu validated CFD solvers against experimental data, before utilizing the solvers to study wave energy converters.

The experimental data used in the validation study described in this report, was gathered by Dixon et al. [19] and is the same data utilized by Westphalen and Hu. Dixons experimental results are chosen for this validation study because of its highly detailed description of the experimental results, including detailed time histories of the forces as well as numerous variations of the experiment parameters.

6.1.1 Dixon et al.'s Experimental Study

Dixon et al. [19] gathered experimental measurements in order to validate a modified version of the Morison equation [27]. In a study preceding Dixons, the wave power research group at the University of Edinburgh's department of mechanical engineering carried out experiments, measuring the wave forces acting on partially submerged fixed cylinders. They found that under certain conditions the resulting force becomes completely negative (downwards), during the entire wave cycle, and

act at twice the wave frequency. These effects arise for bodies close to the surface, due to interaction between buoyancy and inertial forces. Dixon modified the Morison equation to account for these effects and then validated the modified equation against experimental data.

The Morison equation is used to predict loads on fixed bodies in oscillatory flow. The contributions from inertial forces and velocity dependent drag forces are superposed, using empirically determined coefficients to determine the contribution of each.

$$F_h = C_M \rho V \frac{\partial U_h}{\partial t} + \frac{1}{2} C_D \rho A |U_h| U_h \quad (6.2)$$

In the original formulation of the Morison equation (6.2), F_h is the horizontal force, acting on a vertical pile of volume V , ρ is the fluid density, C_M is the inertia coefficient, U_h is the horizontal fluid velocity, C_D is the drag coefficient, A is the projected area of the pile normal to the waves and t represents time.

Dixons modifications to this equation involve adding a buoyancy dependent term and a time dependent volume, to account for the cylinder being partially submerged. For large object to wave amplitude ratios, viscous effects are negligible, and the drag term can be removed.

Dixon introduced a series of dimensionless parameters:

$$F' = \frac{F}{\rho g \left(\frac{\pi D^2 l}{4} \right)} \quad (6.3)$$

$$\eta'(t) = \eta(t)/D \quad (6.4)$$

$$a' = a/D \quad (6.5)$$

$$L' = L/D \quad (6.6)$$

$$d' = d/D \quad (6.7)$$

The relative force F' in equation (6.3) is a measure of the ratio of the wave force F to the weight of water displaced by a totally submerged cylinder, $\rho g (\pi D^2 l / 4)$, where D is the cylinder diameter and l is the cylinder length. Equation (6.4) describes the relative wave elevation $\eta'(t)$, where $\eta(t)$ is the time dependent wave elevation. The relative wave amplitude a' is described in equation (6.5), where a is the wave amplitude. Equation (6.6) describes the relative wave length L' , where L is the wave length. The relative cylinder axis depth d' described in equation (6.7), where d is the distance between the cylinder axis and the still water level.

To validate the accuracy of the modified Morison equation, experimental measurements were gathered using the same parameters as used in the calculations. The values of the parameters a' and d' were varied, but chosen so that part of the cylinder was always exposed to air at some point of a wave cycle. Reflections from the tank end were less than 5% and the force measurements were accurate to within 1% of the largest force measured. The initial force on the cylinder in still water was electronically subtracted before the force measurements were taken.

Dixon found that cases with high relative wave amplitude a' , are the most difficult to reproduce analytically, since they exhibit a high degree of asymmetry. The agreement between the modified Morison equation and the experiments were considered good over a wide range of parameter values. As the relative wave amplitude a' and relative depth d' increased the agreement worsened. The experimental measurements showed an asymmetry, especially in the crest for high relative- amplitudes and depths, which was not predicted by the equation.

6.1.2 Westphalen et al.'s CFD Solver Validation

Westphalen et al. [20] utilized Dixon et al.'s [19] experimental measurements in order to validate four different CFD codes, and then used these codes to analyze a wave energy converter. The codes used were: Smooth Particle Hydrodynamics (SPH), a Lagrangian CFD technique not requiring a mesh, a Cartesian Cut Cell method based on an artificial compressibility method with shock capturing for the interface (AMAZON), and two pressure based Navier-Stokes codes, Finite Volume (FV) and Control Volume Finite Element (CV-FE).

The force measurements from the simulations were non-dimensionalised according to equation (6.3), and then compared with Dixons experimental data. This was performed for three cases, each with different wave amplitudes and axis depths. First order regular waves were generated and the air velocities were set to zero. The results agreed well, except for the case with the deepest cylinder submergence.

6.2 OpenFOAM Numerical Wave Tank Validation

Modeling physical phenomena involves a series of assumptions and estimations. Comparing force measurements from simulations with experimental measurements, like those gathered by Dixon et al. [19], is a method for validating that the implemented models correctly simulates reality.

In cases with low Keulegan-Carpenter numbers N_{KC} , see Table 1, viscous effects are negligible. Thus laminar flow can be assumed and no turbulence modeling is required for this study.

6.2.1 Motivation of Chosen Case Setups

Westphalen et al. [20] validated four different CFD codes against Dixons experimental data. Dixon published experimental data for 24 different cases, and Westphalen selected three of them to validate against. All three cases include the highest possible relative wave amplitude a' available, for three different relative cylinder axis depths d' . The axis depths are equidistantly spaced, with the smallest available axis depth for the first case and the largest available axis depth for the last case.

According to Dixon cases with high relative amplitude a' should offer more of a challenge than cases with low relative, and thus be ideal for validation purposes.

Different levels of relative axis depths d' provide different ratios of inertial and buoyancy forces. For shallow axis submergences, the resulting force display a highly alternating direction, while higher submergence demonstrates a mostly negative (downward) force at twice the frequency. Therefore validating against cases with different axis submergences should provide an indication of the solver capability to deal with different ratios of buoyancy and inertial forces as well as the interplay between them.

For the reasons described above, the three cases chosen by Westphalen could be considered the best cases available for validation purposes. Hence the same three cases used by Westphalen are also utilized in this validation study. Comparison with Westphalens results also provide a reference for what level of agreement to expect. Also, it enables comparison of the OpenFOAM solver not only against experimental results, but also against other CFD codes. See Table 1 for the relative- amplitudes a' and axis depths d' of the selected cases.

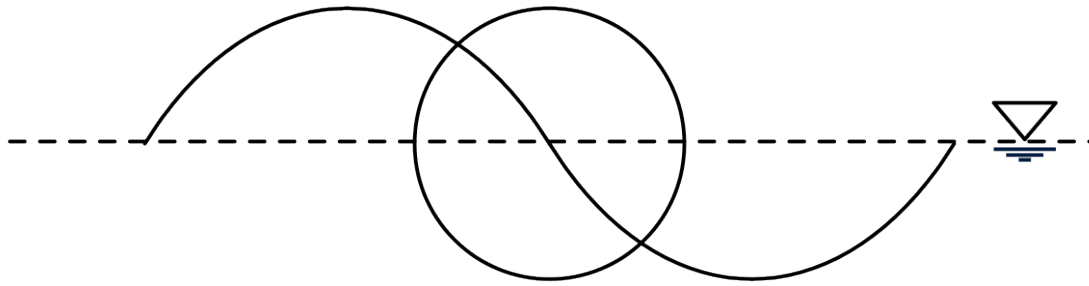
6.2.2 Wave Parameters

The basin is initialized as a still surface and a fifth order Stokes wave is produced in the generating zone. In the damping zone at the opposite end of the basin, the resulting waves are blended with a flat wave. As described in Section 5 the wave properties are specified in the `wavesDict` dictionary and each wave type is specified in separate sub dictionaries. See Table 1 for the wave parameters used in each simulation.

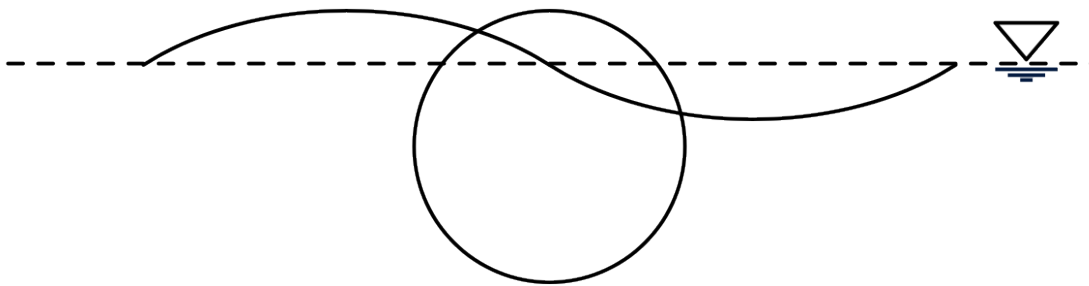
In Table 1, the parameters d' and a' are calculated according to equations (6.7) and (6.5) respectively, with a cylinder diameter of 1 dm. The values are identical to those used by Westphalen [20]. A graphical representation of the relative size of cylinder diameter, axis depth and wave amplitude is seen in Figure 5. Note that the wave length has been shortened in these figures. Since the wave amplitude vary between the cases, a slightly different frequency is required for each simulation in order to achieve a wavelength, λ , of exactly 1.5 meters. The wave steepness's are well below the breaking limit stated by Fenton [2], see equation (2.9). See equation (6.1) for a description of the Keulegan-Carpenter number N_{KC} .

Table 1 Wave parameters used in each of the three simulations

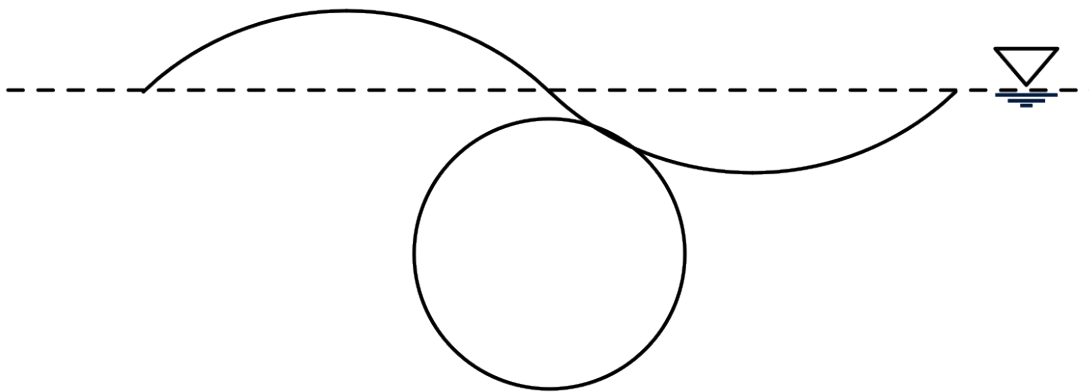
	Simulation		
	1	2	3
Relative axis depth, d'	0.0	0.3	0.6
Relative wave amplitude, a'	0.5	0.2	0.3
Wave frequency, f	1.042	1.023	1.028
Wavelength, λ [m]	1.5	1.5	1.5
Wave steepness, $2a/\lambda$	0.067	0.027	0.040
Keulegan-Carpenter number, N_{KC}	3.142	1.256	1.885
Dimensionless wave amplitude, ka	0.209	0.084	0.126
Dimensionless basin depth, kh	3.770	3.770	3.770



(a) Simulation 1. $d' = 0.0$, $a' = 0.5$



(b) Simulation 2. $d' = 0.3$, $a' = 0.2$



(c) Simulation 3. $d' = 0.6$, $a' = 0.3$

Figure 5 Cylinder axis submergence and wave amplitude for the three simulations

6.2.3 Basin Layout

Both the generating and damping zones have an extension of one wavelength, equal to 1.5 meters. The extension of the unenforced wave tank is two wavelengths, or 3 meters, with the cylinder placed in the center. The zone layout is specified in the `waveGeneratorsDict` dictionary as described in Section 5. Blending of velocity and phase is turned on, while no blending is performed for the pressure field. Figure 6 depicts the layout of the various zones.

The boundary conditions are set to no-slip impermeable walls for the ends as well as at the bottom of the basin. The top is specified as an outlet while the front and back are set to empty, as is the convention for front and back planes in 2D simulations in OpenFOAM. See Figure 6.

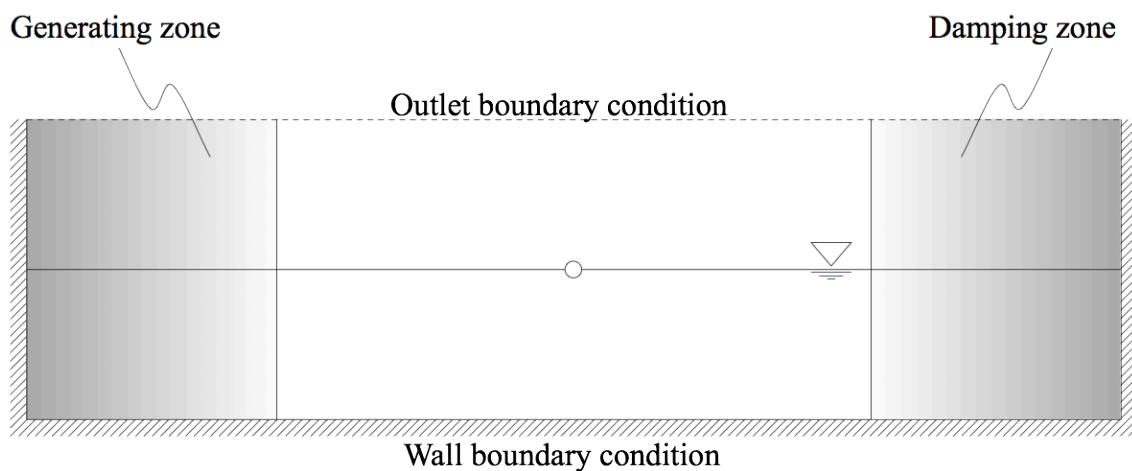


Figure 6 Basin blending zone layout and boundary conditions

6.2.4 Mesh Generation

The meshes are generated using the `blockMesh` utility supplied with OpenFOAM, see Section 4.1.1. Using the macro processing language `m4` for parametric refinement and `python` for calculations, the mesh is generated from a few input parameters, see Table 2.

The cpw parameter determines the basic cell size from which all other cell sizes are calculated. The f_z parameter determines the height of the refined region around the surface. The minimum height of the refined area is equal to the cylinder diameter, and is further increased symmetrically around the cylinder if any part of the wave extends vertically beyond the limits of the cylinder. An f_z value of 1.2 yields a refined region 20% higher than the wave height and cylinder.

These mesh parameters results in a cell count of 234200, 234200 and 335092 cells for simulation 1, 2 and 3 respectively. The meshes for simulation 1 and 2 have an equal number of cells since the wave does not extend above or below the cylinder, while the mesh for simulation 3 has a higher cell count since the wave extends above the cylinder. A close up of the region around the cylinder for simulation 1 can be seen in Figure 7. A band along the surface has been refined and above and below this band the cell size is gradually stretched outwards. The mesh is finest at the cylinder surface and gradually stretched in the radial direction.

Table 2 *Input parameters for mesh generation*

	Simulation		
	1	2	3
Basin length, x-direction, L [m]	6	6	6
Basin width, y-direction, W [m]	1	1	1
Basin height, z-direction, H [m]	1.8	1.8	1.8
Cylinder axis x-displacement, d_x [m]	0	0	0
Cylinder axis z-displacement, d [m]	0	-0.03	-0.06
Cylinder diameter, D [m]	0.1	0.1	0.1
Wave amplitude, a [m]	0.05	0.02	0.03
Cells per wavelength, cpw [cells/ λ]	400	400	400
Refined region height factor, f_z	1.2	1.2	1.2

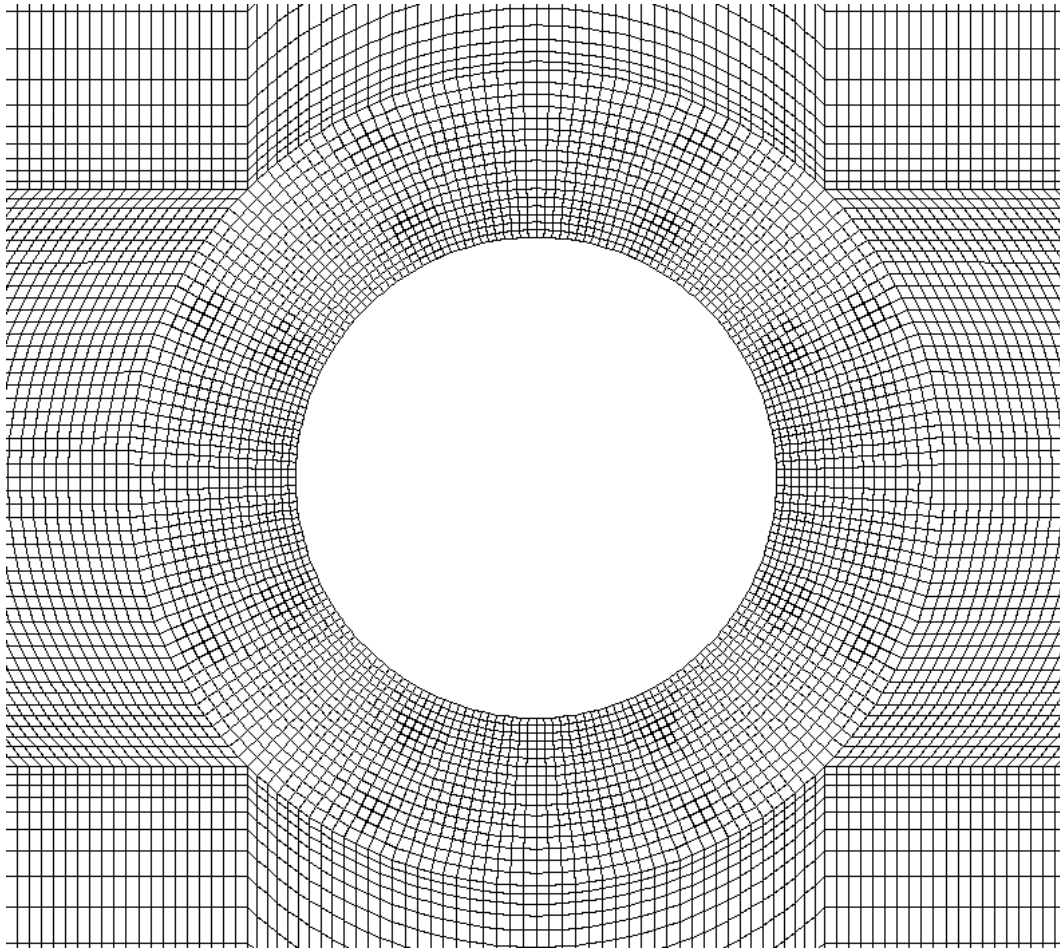


Figure 7 Close up of the mesh around the cylinder for simulation 1

6.2.5 Force Measurement

The forces on the cylinder are registered using a force function in the `controlDict` dictionary. This method of logging forces has the benefit of saving measurements for every time step, not only those matching the specified `writeInterval`, the setting determining at which time steps to save the complete solution. Thus using the force function yields a high measurement resolution without requiring as much space as saving the complete solution.

This function logs the forces and moments at the cylinder surface for each time step, in all three directions. For this particular case only the vertical forces are of interest. Using shell scripting the parameters of interest, time and vertical forces, are extracted and the forces are non-dimensionalised according to equation (6.3). Finally the normalized force time history is imported into Matlab for plotting. See Figure 8 for a flow chart depicting the work flow.

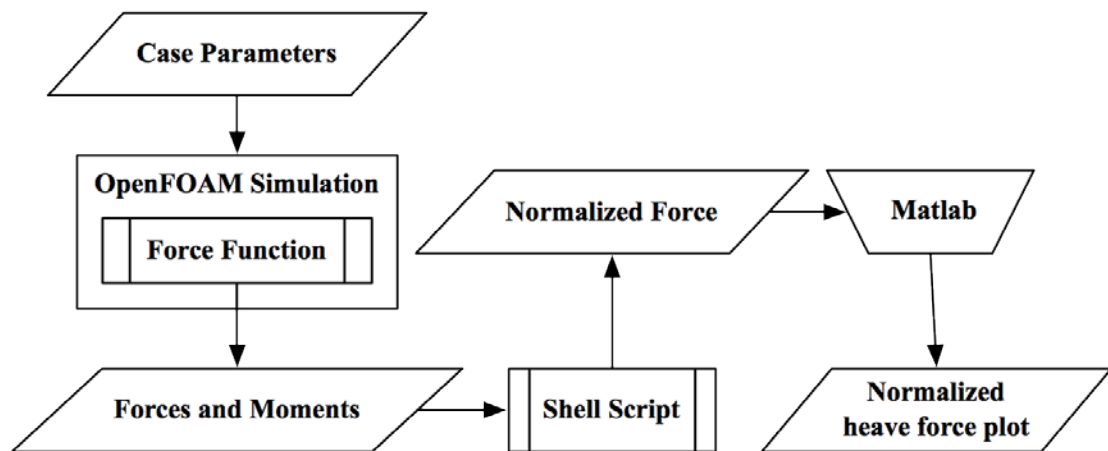


Figure 8 Flow chart depicting the work flow

6.3 Results and Discussion

Figures 9-11 depicts force measurements on semi submerged fixed horizontal cylinders. Information about the simulation setup for the OpenFOAM simulations is found in Section 6.2. The AMAZON, CV-FE and FV results are extracted from simulations performed by Westphalen et. al. [20], see Section 6.1.2. The experimental data was gathered by Dixon et. al. [19], see Section 6.1.1.

The main difference between the simulations is the relative cylinder axis depth d' and the relative wave amplitude a' , see equations (6.7) and (6.5) respectively. The values of these parameters are found at the top of each figure, and also in Table 1 along with other wave parameters. In order to vary the axis depth and wave amplitude, the mesh of each simulation differs slightly, as described in Section 6.2.4.

The vertical axis shows the relative force F' in the vertical direction, calculated according to equation (6.3). The horizontal axis shows the time t divided by the length of one wave period T . Hence one unit distance on the horizontal axis is equal to the time required for one wave cycle to pass the cylinder. C and T in the bottom views indicate the positions in time of crest and trough respectively.

In the top view of Figures 9-11 the force time history of the simulations initialization and the subsequent ten wave cycles are depicted. The square outlines the wave cycle enlarged in the bottom right view. The marked cycle is chosen because it best matches Dixon et. al.'s experimental results.

To better show the agreement between cycles, the bottom left view of Figures 9-11 depicts the ten first wave cycles superposed. All wave cycles should ideally be identical since the simulations lack viscosity. As can be seen this is however not the case for this series of simulations. Possibly the damping zone is too short or the blending function too steep, causing some reflections.

In the bottom right view results from the OpenFOAM simulations are superposed with Dixon et. al.'s experimental data and Westphalen et. al.'s simulation results. A good agreement between simulation and experimental results indicates that the generated waves exerts realistic forces on the cylinder, and thereby to some degree that the underlying models accurately represents reality.

It is hard to quantify the agreement between experimental and simulation results by mere visual comparison. Westphalens results are included as a reference. They provide perspective on which degree of agreement to reasonably expect. From the CFD codes used by Westphalen, the CV alternative should be the one most similar to the OpenFOAM code used in this report.

Simulation 1, depicted in Figure 9, is arguably the least difficult case to simulate. The wave amplitude is high, yielding high force measurements. The cylinder is placed in the center of the still water surface where it is best exposed to the wave forces, further increasing the amplitude of the force measurements. However, the wave is relatively steep, see Table 1. Steep waves generally are more challenging to simulate.

Simulation 2, depicted in Figure 10, has a lower wave amplitude and a deeper submerged cylinder than simulation 1. Thus the resulting force measurement has significantly lower amplitude. Aside from for the somewhat high trough the agreement is considered good.

Simulation 3, depicted in Figure 11, is arguably the most difficult of the three cases to reproduce. Dixon et. al.'s modified Morison equation had troubles recreating it due to the asymmetry. Also Westphalen et. al. failed to achieve good agreement. The OpenFOAM results exhibit some of the same deficiencies as Westphalens, the force curve from the simulation is generally situated higher than the experimental curve. However, the overall shape of the curves agrees well.

Dixon et. al. neglects to mention how the presented experimental results were selected. Similar experiments are presented by Chaplin [24, 25], and several subsequent wave cycles are published. These results indicate that variations may exist between wave cycles for this type of experiment. Thus, evaluating the simulation results by merely comparing them to a single experimental wave cycle is not an ideal validation method.

The best results in the presented OpenFOAM simulations are often found among the first ten wave cycles. Later waves show signs of degradation. As discussed in Section 5.2, several complications were encountered during the simulations, all contributing to the degradation of the results.

Figure 12 shows a comparison of surface profiles between Westphalen et. al.'s simulation and the corresponding OpenFOAM simulation, for subsequent time steps. The surface profiles agree well.

In summary, the simulation results agree well with the experimental results. Zero crossings occur at roughly the same positions and peaks and troughs have approximately the same amplitude. Even smaller peaks and troughs tend to occur at the same positions in simulation and experiment. Whether the agreement is sufficient or not depends on the intended application. A more quantitative judgment regarding the curves agreement would require a more sophisticated method of error evaluation than visual inspection.

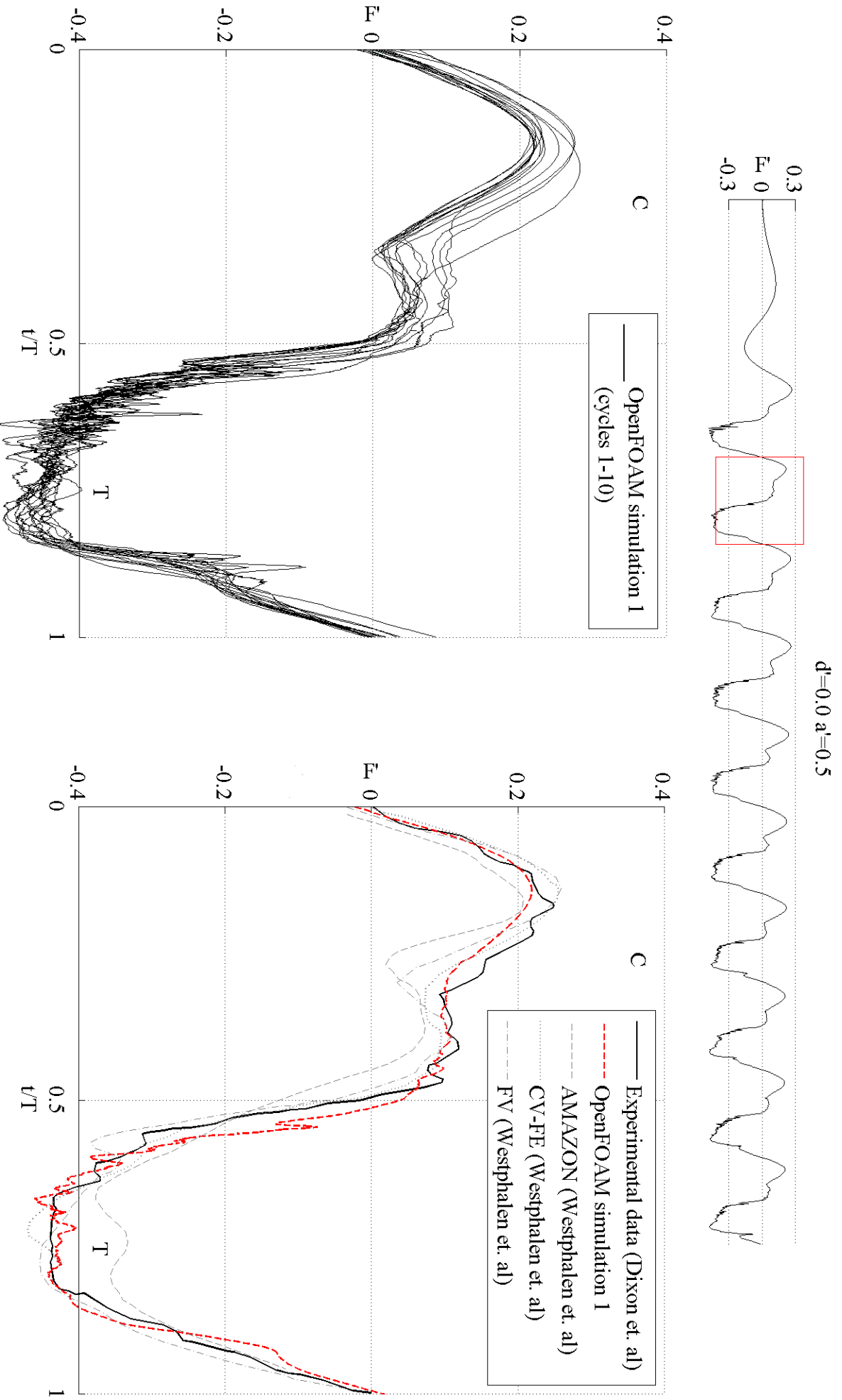


Figure 9 Top: time force history for the first cycles of simulation 1. Bottom left: the ten first wave cycles superposed. Bottom right: the second wave cycle, also marked in the top plot, superposed with Dixon et. al.'s experimental results and Westphalen et. al.'s simulation results.

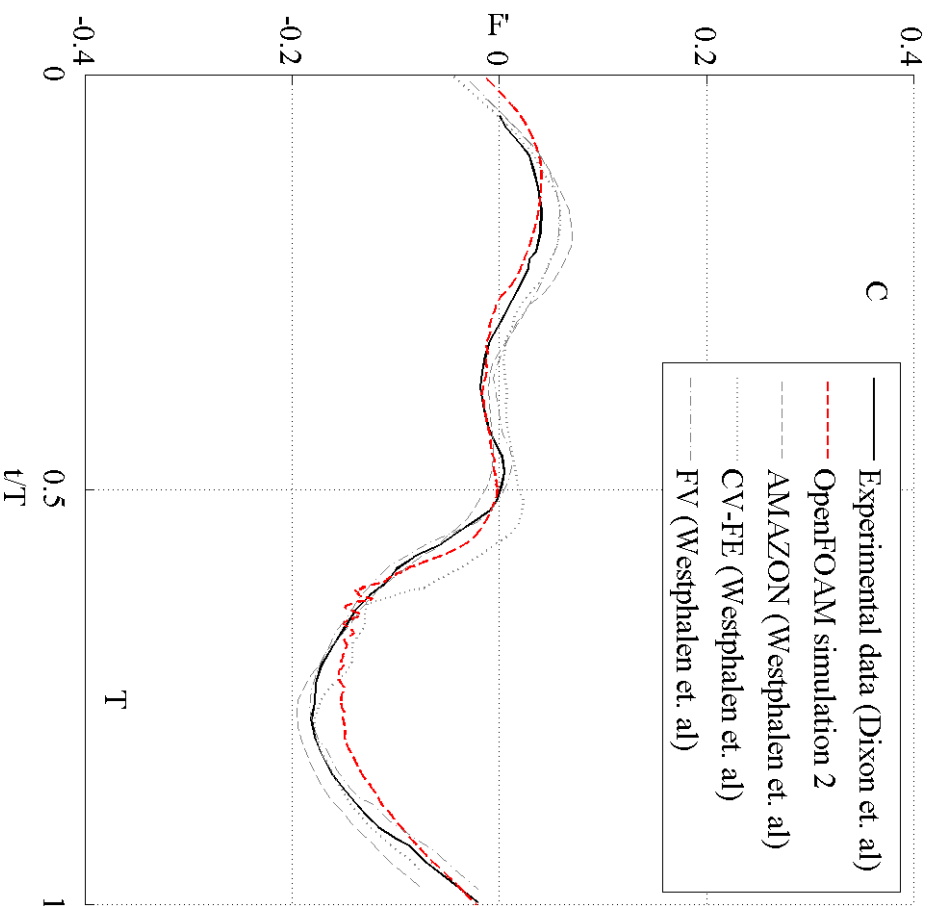
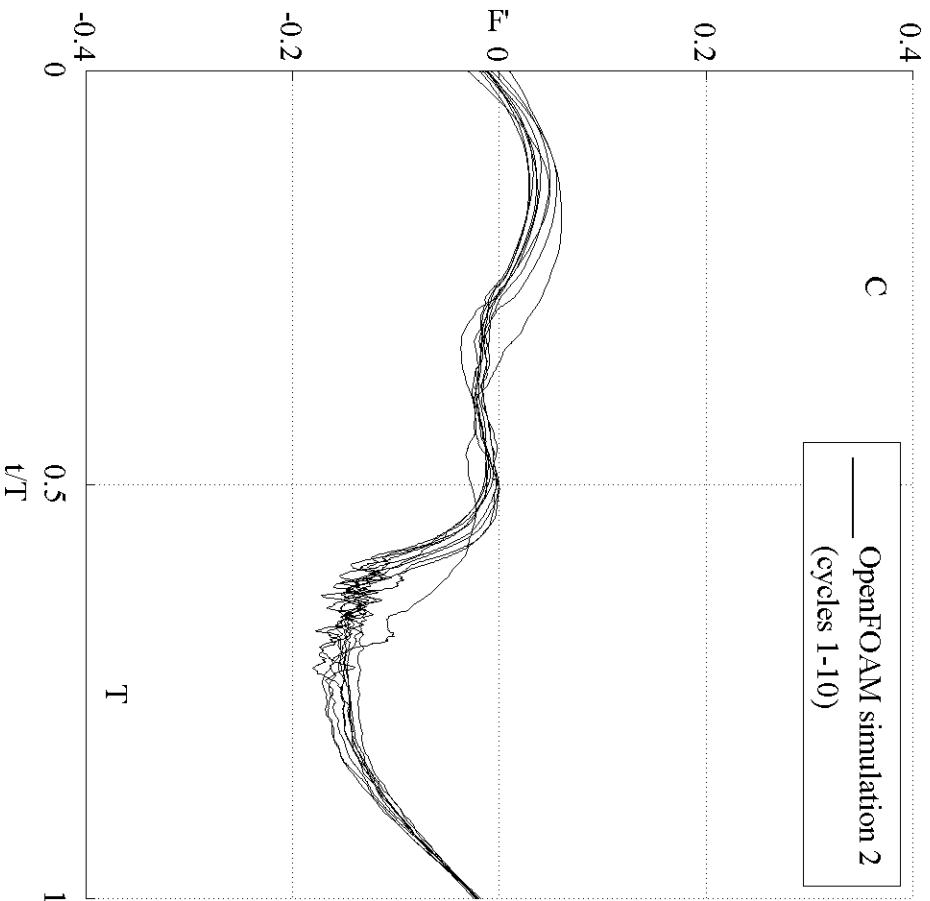
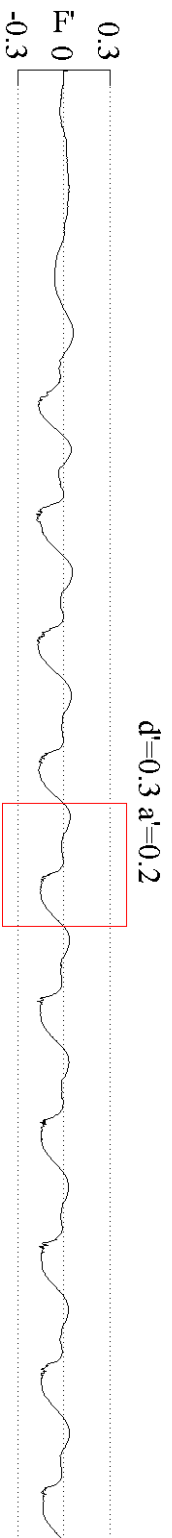


Figure 10 Top: time force history for the first cycles of simulation 2. Bottom left: the ten first wave cycles superposed. Bottom right: the second wave cycle, also marked in the top plot, superposed with Dixon *et. al.*'s experimental results and Westphalen *et. al.*'s simulation results.

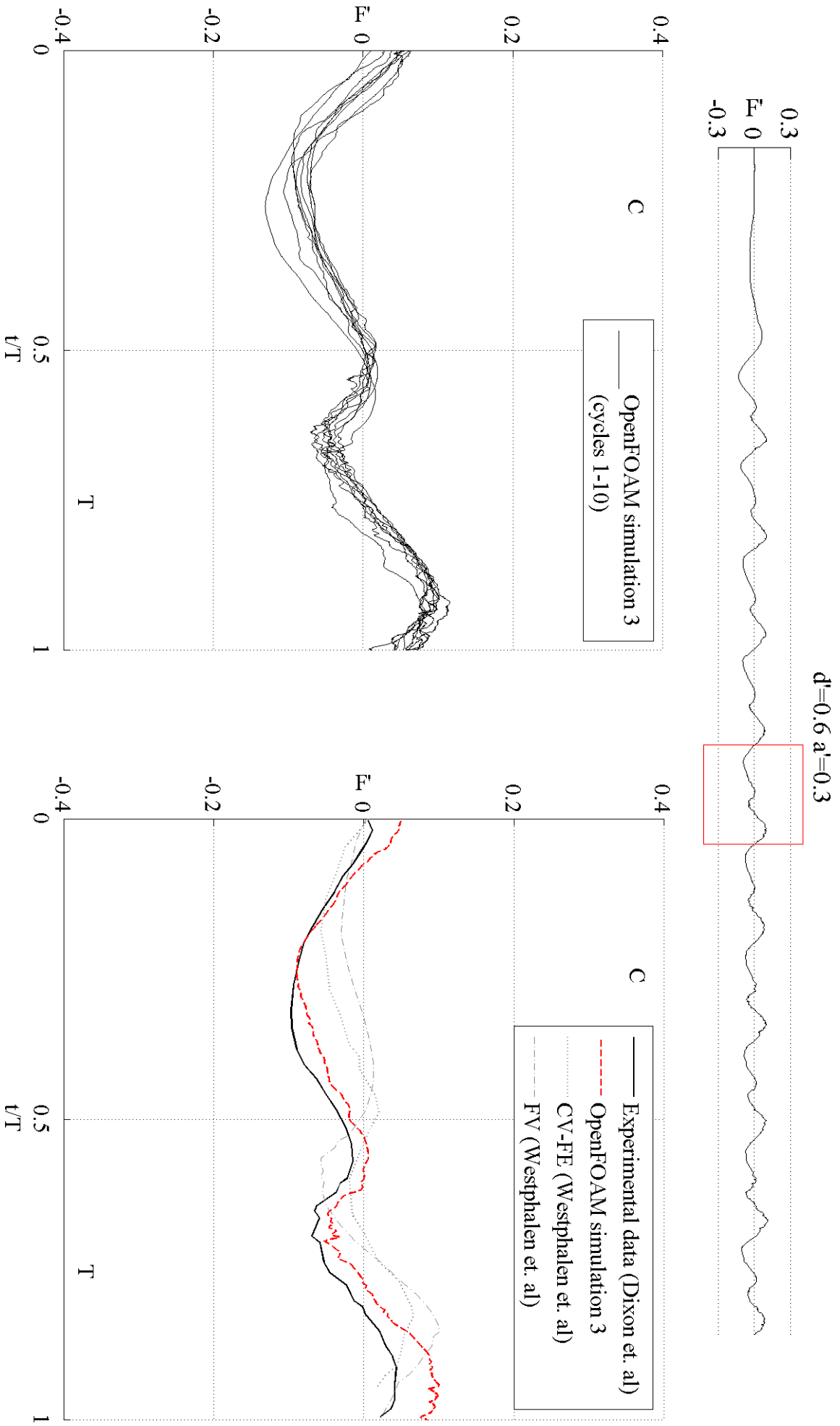


Figure 11 Top: time force history for the first cycles of simulation 3. Bottom left: the ten first wave cycles superposed. Bottom right: the second wave cycle, also marked in the top plot, superposed with Dixon et. al.'s experimental results and Westphalen et. al.'s simulation results.

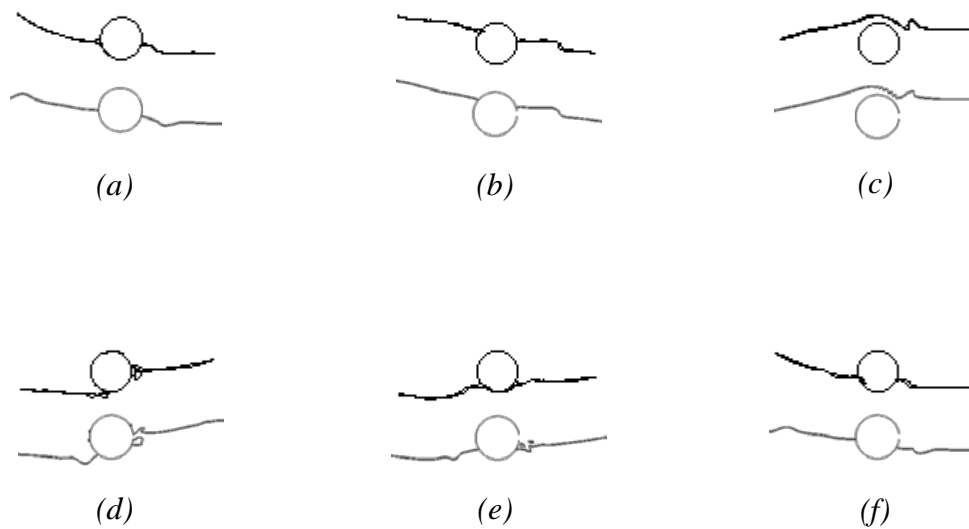


Figure 12 Surface profile around the cylinder for simulation 1, $d' = 0.0, a' = 0.5$. Top: from Westphalen et. al. Bottom: from OpenFOAM simulation. (a) $t/T = 0.0$, (b) $t/T = 0.12$, (c) $t/T = 0.36$, (d) $t/T = 0.6$, (e) $t/T = 0.73$, (f) $t/T = 1.0$.

A grid convergence verification study was attempted, however due to unexpectedly high errors and lack of convergence with decreasing spatial and temporal discretization this study could not be completed. The cause of the unexpectedly high errors is as of yet unknown.

The satisfactory agreement found in the validation study between the OpenFOAM results and the experimental data might appear contradictory to the problems encountered in the verification study. It begs the question if force measurements is a too relaxed method of validation for a wave generator. Part of the answer lies in that the cylinder is placed relatively close to the generating zone, thus not allowing much time and distance for the wave to degrade before reaching the cylinder.

The forces are measured on the entire cylinder surface, a relatively large target compared to the amplitude of the wave. The resulting force is rather a representation of the waves energy, the magnitude and direction of the velocity field, and less sensitive to minor changes in the shape of the surface profile.

During the cylinder validation study, simulation 3 exhibits the worst agreement with the experimental results. The reason for this might be that it is the case which has the deepest cylinder submergence. The cylinder is completely submerged during large parts of the wave cycle. Raising of the trough would have a large effect on how much of the cylinder is uncovered. Due to the difference in density between water and air, most of the forces experienced by the cylinder is due to the water. Thus altering how much air versus water the cylinder is exposed to should significantly affect the experienced forces.

7 Conclusions

This project is performed with the intention of developing a numerical wave tank in OpenFOAM, utilizing the blending method for generating and damping waves. The wave tank is evaluated with a focus on its potential usefulness within SBMs industrial projects.

The implemented blending method is thoroughly integrated into OpenFOAM according to the programming guidelines. This makes it simple to use even for more advanced blending zone layouts, and easily extendable with additional blending functions and wave models.

One of the advancements made during the project was moving to the semi implicit solver. The fully explicit solver yielded severe problems with unrealistic high air velocities while the implicit solver resulted in issues with interface diffusion. The semi implicit solver rectifies both these problems.

The wave induced forces measured on the semi submerged horizontal cylinders agrees well with experimental data and performs on par with other CFD codes. However, some problems still remain. Severe surface profile deformations hindered completion of the grid convergence study. Before the cause of these problems are identified and corrected the potential usefulness of the code in an industrial project is limited.

Additionally, the current implementation is rather slow. This is mainly because such a refined mesh is required in the surface region in order to maintain a sharp interface. This is further aggravated by the blending methods need for large generating and damping zones. Presently no attempts have been made to optimize the codes speed. Possibly implementing an improved surface capturing method, e.g. the level set method, could drastically decrease the required mesh density.

8 References

1. OpenFOAM extended, <http://sourceforge.net/projects/openfoam-extend>, 2011.
2. Fenton, J. D. (1985), *A Fifth-Order Stokes Theory for Steady Waves*. Journal of Waterway, Port, Coastal and Ocean Engineering, vol. 111, no. 2, pp. 216-234.
3. Bergdahl, L. A. (2008), *Wave-Induced Loads and Ship Motions*, Department of Civil and Environmental Engineering, Chalmers University of Technology, no: 2008:1: Göteborg.
4. Fenton, J. D. (1990), *Nonlinear Wave Theories*. The Sea, vol. 9, pp. 3-25.
5. Versteeg, H. K. and Malalasekera, W. (2007), *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*: Prentice Hall.
6. OpenCFD Ltd., <http://www.openfoam.com/>, 2011, The Open Source CFD Toolbox and OpenFOAM User Guide.
7. Jasak, H., Jemcov, A., and Tukovi'c, Ž. (2007), *OpenFOAM: A C++ Library for Complex Physics Simulations*, in *International Workshop on Coupled Methods in Numerical Dynamics*: Dubrovnik, Croatia.
8. CFD Online, <http://www.cfd-online.com>, 2011, An Online Center for Computational Fluid Dynamics.
9. GNU GPL 2, <http://www.gnu.org/copyleft/gpl.html>, 2011, GNU General Public License.
10. Paterson, E. G. (2008), *Multiphase and Free-Surface Flow Simulations*. in *3rd OpenFOAM Workshop*. Politecnico di Milano, Milan, Italy: Applied Research Laboratory and Department of Mechanical Engineering, The Pennsylvania State University.
11. Seindal, R. (2006), *GNU M4, Version 1.4. 7, A Powerful Macro Processor*.
12. Kitware Public, http://www.itk.org/Wiki/ParaView/Users_Guide/Table_Of_Contents, 2011, ParaView User's Guide (v3.12)
13. Jönsson, P. (2006), *MATLAB-beräkningar inom teknik och naturvetenskap*: Studentlitteratur.
14. Lin, P. P., L.F.L. (1999), *Internal Wave-Maker for Navier-Stokes Equations Models*. Journal of Waterway, Port, Coastal and Ocean Engineering, vol. 125, no. 4, pp. 207-215.
15. Engsig-Karup, A. P. (2006), *Unstructured Nodal DG-FEM Solution of High-Order Boussinesq-Type Equations*, PhD Thesis, Technical University of Denmark, Department of Mechanical Engineering, Section of Coastal, Structural and Maritime Engineering.
16. Afshar, M. A. (2010), *Numerical Wave Generation in OpenFOAM*, MSc Thesis, Chalmers University of Technology, Department of Shipping and Marine Technology.
17. Jacobsen, N. G. (2010), *A Generic Wave Generation and Wave Relaxation Framework*. in *5th OpenFOAM Workshop*. Gothenburg: Section of Coastal,

Maritime and Structural Engineering, Department of Mechanical Engineering, Technical University of Denmark.

18. Rusche, H. (2002), *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*.
19. Dixon, A. G., Greated, C. A., and Salter, S. H. (1979), *Wave Forces on Partially Submerged Cylinders*. Journal of the Waterway Port Coastal and Ocean Division, vol. 105, no. 4, pp. 421-438.
20. Westphalen, J., et al. (2009), *Extreme Wave Loading on Offshore Wave Energy Devices using CFD: a Hierarchical Team Approach*. in *8th European Wave and Tidal Energy Conference*. Uppsala, Sweden.
21. Prasad, S. (1994), *Wave Impact Forces on a Horizontal Cylinder*, University of British Columbia, Department of Civil Engineering.
22. Easson, W. J., Greated, C. A., and Duranni, T. S. (1985), *Force Spectra from Partially Submerged Circular Cylinders in Random Seas*. Journal of Waterway, Port, Coastal and Ocean Engineering, vol. 111, no. 5, p. 856.
23. Kaplan, P. and Silbert, M. (1976), *Impact forces on platform horizontal members in the splash zone*. in *8th Offshore Tech. Conf.* Houston, Texas. pp. 749-758.
24. Chaplin, J. R. (2001), *Non-Linear Wave Interactions with a Submerged Horizontal Cylinder*. in *11th International offshore and polar engineering conferenc*. Stavanger, Norway. pp. 272-279.
25. Chaplin, J. R. and Subbiah, K. (1997), *Large Scale Horizontal Cylinder Forces in Waves and Currents*. Applied Ocean Research, vol. 19, no. 3-4, pp. 211-223.
26. Hu, Z. Z., et al. (2011), *Numerical Simulation of Floating Bodies in Extreme Free Surface Waves*. Natural Hazards and Earth System Sciences, vol. 11, pp. 519-527.
27. Morison, J. R., et al. (1950), *The Force Exerted by Surface Waves on Piles*. Petroleum Transaction, American Institute of Mining, Metallurgical and Petroleum Engineers, vol. 189, pp. 149-154.