Implementation of an OpenFOAM Numerical Wave Tank for Wave Energy Experiments

Josh Davidson
Centre for Ocean Energy Research,
National University of Ireland Maynooth
E-mail: josh.davidson@eeng.nuim.ie

Marie Cathelain
Ecole Centrale de Nantes
Nantes, France
E-mail: marie.cathelain@ec-nantes.fr

Louise Guillemet
Ecole Centrale de Nantes
Nantes, France
E-mail: louise.guillemet@eleves.ec-nantes.fr

Thibault Le Huec
Institute of Engineering Sciences of Toulon and the Var
Toulon, France
E-mail: tlehuec@orange.fr

John V. Ringwood
Centre for Ocean Energy Research,
National University of Ireland Maynooth
E-mail: john.ringwood@eeng.nuim.ie

***Abstract*—A numerical wave tank (NWT) can be a useful tool for wave energy experiments. This paper outlines the implementation of a NWT using the open-source computational fluid dynamics (CFD) software, OpenFOAM. In particular, the paper focusses on an NWT designed for experiments involving rigid-body type wave energy converters (WECs), using OpenFOAM version 2.3.0.**

***Index Terms*—Numerical wave tank, CFD, OpenFOAM, wave energy**

I. Introduction

A NWT is the generic name of numerical simulators for modelling nonlinear free surface waves, hydrodynamic forces and floating body motions [1]. NWTs can be useful tools in the design, analysis and optimization of WECs. Current day computing power allows the implementation of NWTs using CFD.

Commercial CFD software packages are available for users and provide helpful user guides, manuals, documentation, graphical user interfaces (GUIs) and real-time customer support. However, the price of these software packages can be prohibitively expensive. Alternatively, there exist open source CFD software packages, free for users, but without the extensive support and documentation of their commercial counterparts.

This paper outlines the development of a NWT implemented using the open source CFD platform OpenFOAM, and aims to act as a helpful guide for researchers wishing to utilise OpenFOAM for the implementation of a NWT, with particular focus on wave energy applications. A brief outline of CFD is given in Section I-A and the choice of the OpenFOAM software explained in Section I-B. A literature review relating to the use of OpenFOAM for implementing NWTs is collated in Section II. Examples of the different types of wave energy experiments and the general capabilities of the NWT are outlined in Section III. The remainder of the paper then details the implementation of the NWT for wave energy experiments using the OpenFOAM software.

A. Computational Fluid Dynamics

The dynamics of fluids is governed by the transfer of mass, momentum and heat, which are described by the Navier-Stokes equations [2]. In general, these equations have no known analytical solution, however, they may be solved numerically using CFD by discretising the domains of space and time to form a system of linear algebraic equations, which are computer implementable.

CFD treats the fluid-structure interaction problem, using the scheme outlined in Figure 1. First, the Navier-Stokes equations are solved for the fluid pressure and velocity throughout the domain. The fluid pressure is then integrated over the body's surface, to give the hydrodynamic force on the body. The resulting body motion, due to the hydrodynamic force, is then calculated using Newton's laws. The body and the fluid states are updated, the simulation is iterated forward to the next time step and the process is repeated.

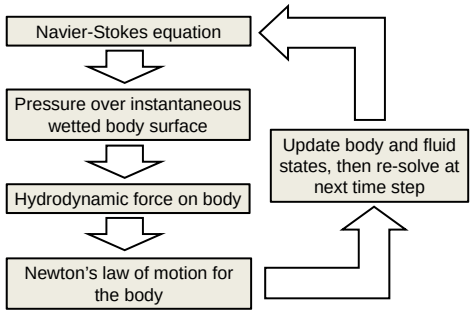

Fig. 1. Schematic of CFD process for fluid-structure interaction

B. OpenFOAM

There are numerous CFD software packages capable of implementing a NWT. The choice of OpenFOAM for the present application is due to its open-source licensing, which allows the user to run as many jobs/tasks as they require on an unlimited amount of processors, for free. Whereas the number

 Copyright © European Wave and Tidal Energy Conference 2015

of tasks and processors available for commercial codes depends upon license fees. Furthermore, OpenFOAM is designed to run on open-source Unix/Linux systems, which eliminates any software costs entirely from the NWT implementation. The remaining costs for performing the NWT experiments are then: the man hours for setting up and post-processing the simulations, and the computing costs for simulating the experiments, due to hardware procurement, electricity for operation etc, or from renting time on a high-performance computer.

OpenFOAM is essentially a collection of C++ code written in text files. Operating a software package entirely by text based commands, instead of a helpful GUI, may at first seem like a weakness, however it turns out to offer the following strengths:

- No waiting for GUI's to load when launching, nor the inconvenience of the GUI's freezing and needing to be restarted during operation .
- Scripting:
  - Automation scripts reduce the aforementioned man hour costs.
  - Automation is performed by first parameterising the C++ text files. Then for each individual case the parameters in the C++ text files are overwritten with user defined parameter values for different NWT experiment set-ups.
  - The NWT in the current paper uses the open source software *Python* for scripting:
    - Required inputs - STL file of the device geometry and the device mass matrix.
    - User options - Tank geometry, wave inputs, types of experiments , power take-off (PTO) settings etc.
    - Postprocessing, data processing and plotting are also available with *Python*.
- It allows full control over the software, giving the user freedom to modify it to suit their needs. For example: Jacobsen et al. [3] modified a fluid solver to generate and absorb waves and currents, and Palm et al. [4] modified a rigid-body solver to couple it to an external solver for mooring loads.
- OpenFOAM is constantly evolving and improving, as users extend upon the code, implementing libraries and toolboxes for different applications. The open-source nature of OpenFOAM often leads to these useful libraries and toolboxes being freely shared in the public domain. For example: Jacobsen et al. [3] shared the modified fluid solver as a free toolbox, waves2Foam [5], for creating and absorbing waves and/or currents, and Palm et al. [4] publicly shared a step by step guide of how to couple an external mooring solver with OpenFOAM [6].
  It is with this same sense of open-source spirit that the current paper is written, in the hope that other researchers can use, improve upon and then share open source NWTs for wave and other related offshore renewable energy applications.

II. OpenFOAM literature review

OpenFOAM (Open source Field Operation And Manipulation) has been used in a wide range of science and engineering applications, such as complex fluid flows, heat transfer, chemical reactions and electromagnetics. This section collates relevant literature pertaining to the use of OpenFOAM for the application of implementing a NWT for wave energy experiments and is arranged into the following subsections: Introductory documents, wave modelling and coastal processes, wave-body interactions, hydrodynamic parameter identification, WEC simulation, theses, and then finally other related material.

A. Introductory documents

For a novice user the following documents offer a useful introduction to OpenFoam and are helpful in getting started and becoming familiar with the software. The OpenFOAM user guide provides general information on the operation of the software and offers some basic tutorials [7].

Chalmers University of Technology offers a MSc/PhD course in CFD with open source software, where the majority of the course is focussed on OpenFOAM. All of the course material is freely shared online [8]. Likewise, the University of Genoa offers a course in OpenFOAM and generously shares all of the material online also [9].

B. Wave modelling and coastal processes

The use of OpenFOAM to model the propagation and breaking of waves has been reported by a number of authors. Morgan et al. [10] used OpenFOAM to reproduce experimental results for the propagation of monochromatic waves over a submerged bar. Chenari et al. [11] also modelled the propagation and breaking of regular waves. Extreme wave events were investigated by Vyzikas et al. [12] by modelling a focussed wave event, in an experimental wave tank and in an OpenFOAM NWT, whereby they found good agreement between the two results until unwanted wave reflections contaminated the experimental wave tank results, highlighting one of the advantages of the NWT approach for wave energy experiments. OpenFOAM has been also used to simulate coastal processes [3] [13].

C. Wave-body interactions

Li and Lin [14] used a two-dimensional OpenFOAM numerical wave tank to simulate nonlinear wave-body interactions between monochromatic waves and a stationary surface-piercing body in water of finite depth, with flat and sloping bottoms, and found good agreement with experimental and numerical results of other researchers. A similar study was later performed by Li and Lin [15] including irregular waves and varying water depth. Chen et al. [16] assessed how OpenFOAM performs when applied to non-linear wave interactions with offshore structures for a range of wave conditions.

D. Hydrodynamic parameter identification

Bonfiglio et al. [17] performed prescribed harmonic oscillation experiments on a floating body to determine the hydrodynamic radiation coefficients of the body at numerous frequencies using a viscous solver. Present day coefficients were determined using velocity potential methods which exclude viscosity, therefore by performing experiments using OpenFOAM, Bonfiglio et al. were able to measure the hydrodynamic forces on the body during radiation experiments and compare their values to those predicted by boundary element solvers. Cathelain [18], Garrido-Mendoza et al. [19] and Gadelho et al. [20] also repeated this approach to determine the added mass and radiation resistance coefficients of a floating body at discrete frequencies.

Davidson et al. [21] used free decay experiments to identify the full state dynamics of a floating body. Armesto et al. [22] used free decay experiments and input waves to identify the state plus input dynamics of an oscillating water column (OWC). Davidson et al. [23] identified the state plus input dynamics of a floating body by introducing a PTO force to the body to drive its motion. Nonlinear hydrodynamic restoring force coefficients were also identified from the NWT experiments in [23]. Giorgi et al. [24] used input waves to identify nonlinear hydrodynamic excitation force kernals. Ringwood et al. [25] details optimising NWT experiments for the identification of hydrodynamic models and shows examples of both input waves and PTO forces in a NWT experiment to train and validate a generalised hydrodynamic model.

E. Wave Energy Converter simulation

OpenFOAM has been used to simulate the performance of a range of WEC's with different working principles. Generic point absorber type WECs coupled to moorings were modelled by Palm et al. [4] [26]. OWCs have been simulated by Iturrioz et al. [27] and Souza et al. [28]. The oscillating wave surge converter, 'The Oyster', has been modelled in OpenFOAM by Schmitt et al. [29] and Henry et al. [30]. A rotational WEC using a single-bucket drag type turbine was simulated by Akimoto et al. [31]. Thaker and Banerjee [32] simulated two-phase flow phenomena in horizontal pipelines to investigate WECs such as the JOSPA and Vigor. King [33] performed 2D simulations of the membrane based WEC, the 'Bombora'. Overtopping, the working principle for a whole class of WECs, is unable to be modelled using linear theory, therefore Eskilsson et al. [34] used OpenFOAM to numerically simulate the overtopping behaviour of the Wave Dragon device.

F. Theses

There are a number of theses focused on the use of OpenFOAM for hydrodynamic simulations. These useful reference documents, owing to the longer nature of a thesis compared to a paper which allows more detail to be included.

Afshar [35] produced a masters thesis on numerical wave generation in OpenFOAM. Alsgaard's masters thesis [36] investigated piston mode resonance in a moonpool using OpenFOAM. Simulation of wave induced forces on semi submerged horizontal cylinders were performed in Andersson's masters thesis [37]. Lambert [38] outlined the development of a NWT using OpenFoam in her masters thesis. Paulsen [39] used OpenFoam to compute wave loads on offshore strucutres in his PhD Thesis. Cathelain's masters thesis [18] focussed on the modelling of a floating body in heave motion using an OpenFOAM NWT. The use of OpenFOAM for the application of coastal wave/structure interaction was explored in Morgan's PhD thesis [40]. Palm's Liciante Thesis [41] outlined the development of his work into simulating moored floating WECs.

G. Of interest

Thien et al. [42] estimated the hydrodynamic forces on a floating airboat from numerical simulations in OpenFOAM. Benitz et al. [43] used OpenFOAM to analyse the hydrodynamic forces on a semi-submersible offshore wind platform. The hydrodynamic forces on a full scale tension leg platform were investigated using OpenFOAM by Dai et al [44]. Ding et al. [45] investigated energy harvesting from flow induced motion and vortex induced vibrations using physical experiments and OpenFOAM simulations, noting good agreement between the two.

III. OpenFOAM Numerical Wave Tank for Wave Energy Experiments

Here, the use of the general purpose CFD software, OpenFOAM, is presented for the specific purpose of implementing a NWT for wave energy experiments. Section III-A describes using OpenFOAM to create a NWT, Section III-B outlines the OpenFOAM NWT's capabilities and then a number of example wave energy experiment are displayed in Section III-C.

A. OpenFOAM Numerical Wave Tank

The NWT uses the *interFoam* solver, designed to solve the Navier-Stokes equations for two incompressible fluids. Fig. 2 shows a snapshot of a NWT simulation depicting one fluid in red and the other in blue, obviously for wave energy applications the two fluids are sea water and air, respectively.

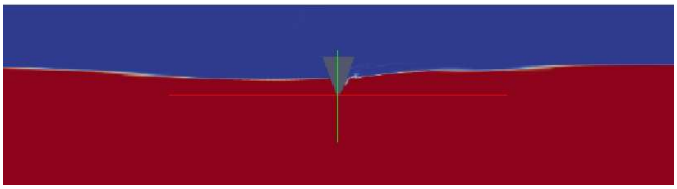

Fig. 2. Snapshot of a NWT simulation

interFoam tracks the free surface using the volume of fluid (VOF) approach, illustrated in Fig. 3 which shows the volume fraction of water inside each mesh cell. It is therefore important to have a fine vertical mesh resolution around the area where the free surface is likely to appear in order

0	0	0	0	0	0	0	0	0
0.07	0	0	0	0	0	0	0	0
0.91	**0.76**	**0.57**	**0.48**	**0.27**	**0.12**	**0.02**	0	0
1	1	1	1	1	1	**0.94**	**0.89**	**0.78**
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

Fig. 3. The volume of fluid technique for capturing the free surface interface. Here the actual free surface is represented by the solid blue line and the number represent the volume fraction of water in each cell

to accurately capture the free surface interface (see Section IV-B3).

interFOAM offers a number of methods and models to simulate turbulence. Turbulence models are used to predict the effects of turbulence in fluid flow without resolving all scales of the smallest turbulence fluctuations. A number of models have been developed that can be used to approximate turbulence based on the Reynolds Averaged Navier-Stokes (RANS) equations.

- Laminar model
- $k-\epsilon$ model
- $k-\omega$ model
- $k-\omega-SST$ model

Within each time-step, *interFoam* therefore not only solves the Navier-Stokes equations for the fluid pressure and velocity, but also solves for the volume fraction and a number of variables associated with the turbulence models.

Different functions and utilities can then be used with the *interFoam* solver to enable various NWT capabilities. A rough abstraction of using OpenFOAM to implement a NWT is depicted in Fig. 4.

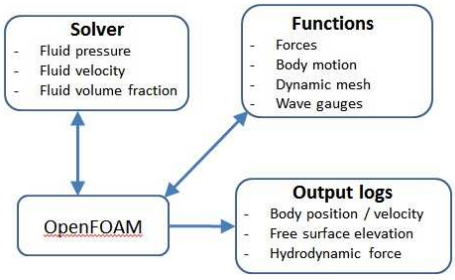

Fig. 4. Using OpenFOAM to implement a NWT.

The *forces* function integrates the fluid pressure over a boundary, such as the surface of a WEC, to calculate the hydrodynamic force on that boundary. In Fig. 5 the fluid pressure over the surface of a sphere at a particular instant is shown. The forces function outputs the calculated hydrodynamic forces and moments in six degrees of freedom (DoF), either to output logs or to other functions or utilities.

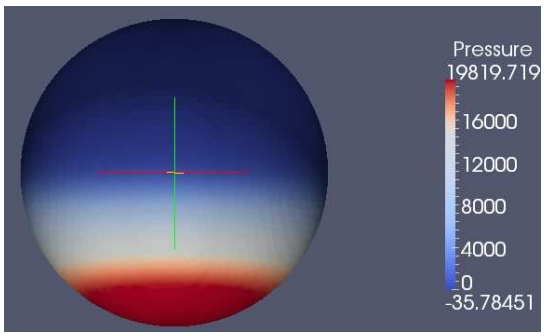

Fig. 5. Fluid pressure over the surface of a semi-submerged sphere

The two-way fluid-strucutre interaction between the fluid and a WEC can be implemented using the *sixDoFRigidBodyMotion* utility. *sixDoFRigidBodyMotion* takes the hydrodynamic force calculated by the *forces* function, then uses Newton's laws of motion to calculate the motion of the WEC. Other functions are then called to update the position of the WEC and the mesh cells. The simulation then iterates forward in time and the process is repeated with the *interFoam* solver calculating the new fluid pressure and velocity throughout the domain. *interFoam* allows dynamic mesh motion using *interDyMFoam*.

B. Capabilities

The OpenFOAM NWT has the following features which are useful for wave energy experiments:

- Can test at full scale, eliminating scaling effects.
- Viscous effects, nonlinear waves, wave breaking, green water effects, slamming loads, etc. neglected by velocity potential methods are included.
- Any tank depth / width can be used and changed as desired.
- Import any WEC geometry without incurring costs involved for building prototypes.
- Wave creation and absorption, offering control of wave inputs not possible in the open ocean testing and eliminating wave reflections which is difficult in physical wave tanks.
- Hydrodynamic force measurement.
- Actuate or constrain motion in six DoF without mechanical restraints influencing device dynamics.
- Passively probe the pressure and velocity values for the air and water everywhere with zero measurement noise.

C. Examples

In this section a sample of various NWT experiments are shown. The scope of the present paper considers rigid body type WECs only. However, the use of OpenFOAM NWTs for OWC applications has been reported in [27] [28] and early demonstrations involving flexible membranes is shown in [33].

Fig. 6 shows a plot of the free surface elevation measured by a wave gauge in the NWT and a plot of the corresponding spectral content. The waves are generated using the methods

outlined in Section IV-C1, targeting to produce a JONSWAP spectrum with a peak period of 10s and consisting of 50 frequencies evenly spaced between 0 and 0.5Hz with randomly assigned phases.

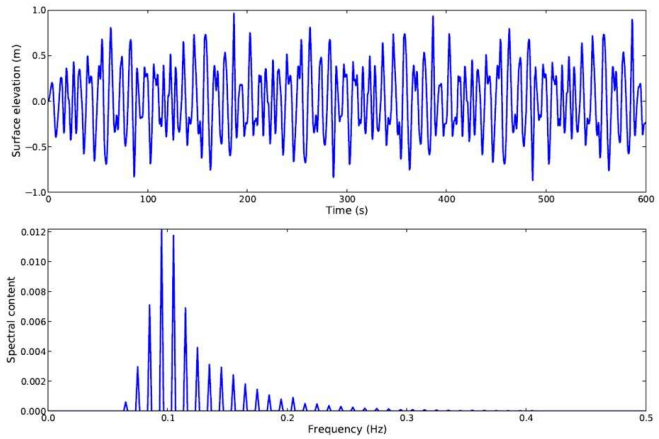

Fig. 6. Irregular waves experiment

A simple example of body motion is the free decay experiment, Fig. 7, whereby the body is initially displaced from equilibrium and the resulting motion simulated as the body oscillates back to its rest position. This experiment is useful for investigating a WEC's natural dynamics, yielding information such as decay rates and system resonances (shown by the peak in the spectral content in Fig. 7).

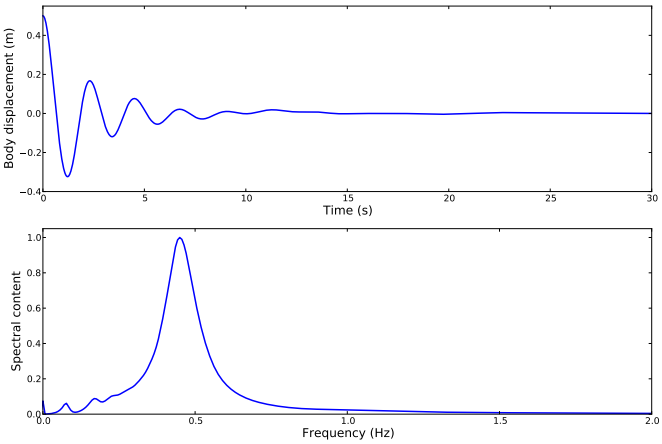

Fig. 7. Free decay experiment

Another useful experiment for examining the hydrodynamic response of a WEC, is to drive its motion with an input force, as shown in Fig. 8. In this figure a power take-off (PTO) system, with bi-directional power flow capabilities, inputs energy to the WEC and drives its motion with the PTO force using a chirp signal. The instantaneous frequency of the chirp signal increases linearly during the simulation, sweeping the range from 0 to 1Hz in 300s for this example. The corresponding body motion gives an indication of the system's response for the corresponding instantaneous input frequency, illustrated by the different resonant peaks in Fig. 8.

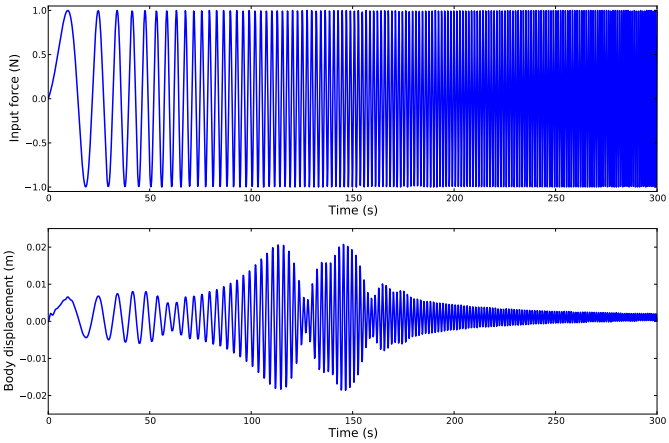

Fig. 8. PTO input force experiment

Fig. 9 shows an example output from an experiment where the WEC motion is excited by an incident sea spectrum. The NWT can generate single or multi-frequency input waves, which can be either omni- or multi-directional, and is also capable of producing winds and currents. For these type of experiments, involving WEC motion driven by input waves, the NWT can also include a PTO force on the WEC. This functionality, in addition to the NWT's ability to instantaneously probe any variable with zero measurement noise and apply perfect actuation forces, makes the NWT a useful testbed for designing and evaluating PTO control algorithms. Additionally in these type of experiments, the NWT can include realistic mooring forces on the WEC, as shown by [26].

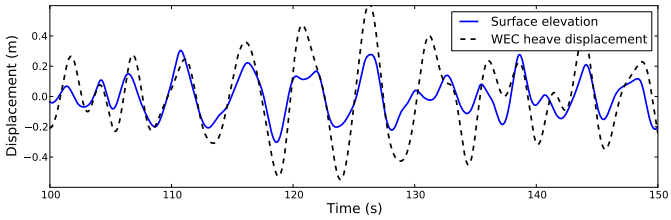

Fig. 9. Input waves experiment

The NWT's ability to measure the hydrodynamic force on a body, offers many useful possibilities in wave energy experiments. Fig. 10 shows an example of an experiment designed to measured the hydrostatic restoring force for a WEC. In this experiment the WEC is slowly moved through its full range of displacement while the corresponding hydrodynamic force is measured, Fig. 10-(a) and 10-(b) respectively. If the WEC is moved slowly enough then all of the velocity and acceleration related effects (dynamic forces) will be negligible and the measured hydrodynamic force will consist only of the position dependent effects i.e. the hydrostatic force. The hydrostatic force can then be evaluated as a function of the WEC displacement, 10-(c), from which nonlinear hydrostatic

restoring force terms can be identified for WEC models [23].

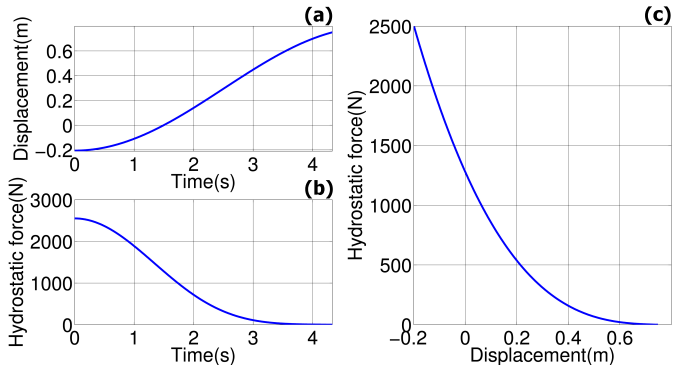

Fig. 10. Hydrostatic restoring force experiment

IV. Implementation

This section details the implementation of the OpenFOAM NWT for the case of a rigid body type WEC. Section IV-A describes the *Case Folder*, where the files for each different OpenFOAM simulation reside. Section IV-B outlines the steps involved in defining and meshing the tank and WEC geometries. Section IV-C focuses on the fluid settings, such as; defining the physical properties of the air and water, the different solver options, the initial and boundary conditions, and how to create and absorb waves. Section IV-D involves implementing the body motion in the NWT. Distributing the simulation across multiple processors for computation is outlined in Section IV-E. Section IV-F then discusses measuring and recording outputs from the NWT simulation. Finally, Section IV-G offers advice on debugging a simulation crash.

A. Case Folder

The files in the *case folder* contain all of the information for an OpenFOAM simulation. The OpenFOAM utilities and solvers are invoked from the *case folder* to pre-process and simulate NWT experiments. The output data is then written to this folder for post-processing. The files organised into three sub-folders:*Constant*, *System* and *0.org*.

1) Constant: The files in the *Constant* folder contain information relating to the physical properties of the NWT, such as: geometry and mesh data, fluid and turbulence properties, direction and magnitude of gravity etc.

2) System: The files in the *System* folder contain information relating to the simulation properties of the NWT, such as: the simulation length, the solvers used, the discretisation methods used, any functions or libraries to be included etc.

3) 0.org: The files in the *0.org* folder contain information relating to the initial and boundary conditions of the solution parameters, such as: the fluid velocity, pressure, volume fraction, location of rigid bodies etc.

B. Tank and device geometry creation

Creating the geometrical properties of the NWT are discussed in this section. The tank's geometrical domain is defined and then names assigned to the relevant boundary faces. A backgrouund mesh is constructed, from which the device geometry is imported into.

1) Tank: The geometrical domain of the simulation is defined by vertex points in the *blockMeshDict* file, which is located in *polyMesh* folder inside the *constant* folder. From these vertices; blocks and edges can be defined, also in the *blockMeshDict* file, to construct the overall geometry of the domain.

The present NWT is constructed using rectangular blocks. The vertex points are given parameterised names (e.g. $x1, x2, .., y1, y2, , .z1$ etc) and then later assigned a numeric value by the *Python* scripts. The parameter values are calculated depending on other experiment parameters selected by the user, such as the depth of the tank, the wavelengths to be generated etc. The domain is split into 27 blocks in a 3x3x3 arrangement, whereby one central block is neighboured on both sides in all directions.

The rectangular blocks allow a clean divide through symmetry planes in the two horizontal directions, for construction of the half and quarter tanks, illustrated in Fig. 11. For experiments which exhibit one or two planes of symmetry, it is beneficial from a computational view to only model one half or quarter of the domain respectively. Symmetry planes can be defined on boundary faces, discussed in the next subsection.

The rectangular blocks also allow an easy construction of a 2D tank. Strictly speaking OpenFOAM does not perform 2D simulations, however, by removing the two outside rows of blocks in one horizontal direction, leaving a 3x1x3 arrangement, and placing symmetry planes on the front and back faces of this domain, will yield a simulation that is only one cell thick and able to run very fast compared to its 3D counterpart. 2D tanks can be useful to save on computation time while developing experiment design.

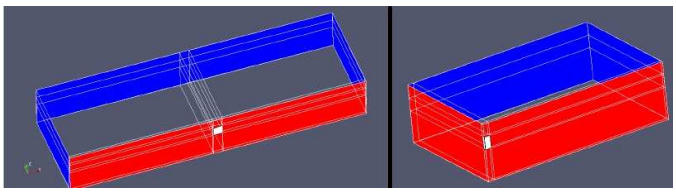

Fig. 11. The half and quarter tanks. Symmetry planes are denoted by red and side walls by blue

2) Boundaries: The boundaries are defined in the *blockMeshDict* file. The tank floor is assigned the *wall* boundary type, whereas the top face and the the four walls of the NWT are of type *patch*. For simulations in the half, quarter or 2D tanks, the boundary type *symmetryPlane* is assigned to the outer vertical faces not corresponding to the actual NWT walls.

3) Background mesh: The mesh is a very important part of the calculation. Poor meshes lead to poor results. There exists a tradeoff between number of cells and computation time. The meshing strategy for the current NWT is described below and illustrated in Fig. 12:

- The central block contains a high density mesh of uniform cubic cells.
- Horizontally, the mesh stretches with distance away from the central block to reduce the overall amount of cells.
- Vertically, the mesh is split into three regions: air, interface and water.
  - The interface region spans the area where the free surface may appear during the simulation and is set with a high mesh density of uniform vertical cell length.
  - The air region has a low cell density and the mesh stretches with distance from the interface upwards to the atmosphere boundary.
  - The water region has moderate mesh density and the mesh stretches gradually from the interface down to the tank floor.

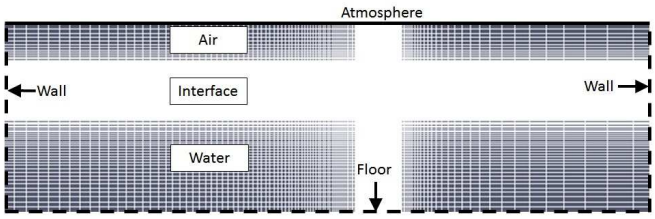

Fig. 12. Background mesh

4) Importing the WEC geometry: The *snappyHexMesh* utility removes and refines cells from the background mesh to import the WEC geometry into the NWT, as depicted in Fig. 13. *snappyHexMesh* is controlled by the *snappyHexMeshDict* file in the *system* folder and requires that the user provide an STL file of the geometry in a folder named *triSurface* in the *constant* folder.

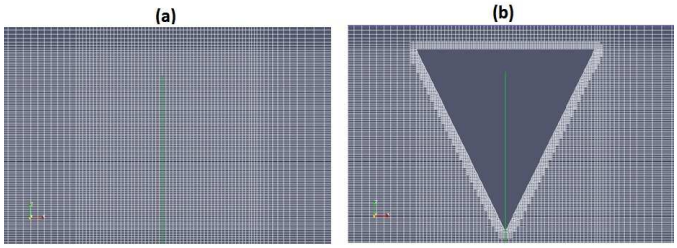

Fig. 13. (a) Original background mesh. (b) Using *SnappyHexMesh* to import a geometry into the background mesh

The geometry is imported into the central, high mesh density, region and *snappyHexMesh* can also be used to further refine the mesh around the WEC to adjust mesh quality parameters to desired levels. For example, in turbulent flows close to boundaries, the mesh's $Y+$ value is important in order to resolve the boundary layer. The $Y+$ value is the nondimensional wall distance;

$$Y+ = \frac{y\sqrt{\frac{\tau_\omega}{\rho}}}{\nu}, \tag{1}$$

here y is the smallest element height normal to the wall, ν is the kinematic viscosity and τ_ω is the wall shear stress. *SnappyHexMesh* can be used to refine the mesh around the device to ensure that these cells obey the relevant $Y+$ values. For some applications this may result in a very large cell count and slow simulations. For this reason [34] advises to run Euler simulations without the viscosity where appropriate.

C. Fluid settings

The NWT is based on *interFoam*, which solves for the fluid pressure, velocity and volume fraction, as well as additional variables in the turbulence models. This section focuses on the treatment of these variables, which correspond to the different file names in the *0.org* folder:

- **u** : the velocity
- **p** : the pressure
- **p_rgh** : the 'dynamic pressure'. Derived from the calculated pressure and is used to simplify the treatment for certain boundary conditions. Specifically, it equals the pressure minus the density of the fluid multiplied by gravity multiplied by the vertical height of the cell from zero. If the free surface is located at zero this quantity would represent the total pressure minus the hydrostatic pressure, but for all other cases this is not true.
- **alpha.water** : the volume fraction of water
- **k** : turbulent kinetic energy
- **epsilon** : turbulent dissipation
- **omega** : specific dissipation
- **nut** : kinematic eddy viscosity

The initial and boundary conditions for these variables are set inside the corresponding variable file in the *0.org* folder. The volume and initial position of the two fluid phases, air and water, are set using the *setFields* utility which reads the *setFieldsDict* in the *system* folder. The properties of the fluids, such as their density and viscosity are set in the *transportProperties* file and the turbulence settings in the *turbulentProperties* file, both located in the *constant* folder.

1) Wave creation and absorption: There are a numerous options for creating and absorbing waves in the NWT ranging from; geometrical beaches, porosity zones, high viscosity regions and cell stretching techniques for absorbing waves, and moving wall/flap type wave makers, dynamic boundary conditions and relaxation zones for both wave creation and absorption. Each have different strengths and weaknesses, a comparison of these different methods is outside the scope of the present paper.

The current NWT uses the relaxation zone method as implemented by the waves2Foam toolbox [3]. This method blends the solutions calculated by *interFoam* with target solutions to produce a target output. The target output for absorption is zero velocity. For wave creation, the user is able to select a target solution from a number available wave theories, set in the *waveParameters* file in the *constant* folder, to produce the desired output. The blending occurs inside of relaxation zones, which are depicted in Fig 14 for the case of a unidirectional wave train travelling from the left to the right of the tank.

Due to the large sizes of these relaxation zones this technique is not likely to be the most computationally efficient method, however the toolbox functionality makes it easy to implement. Initial conditions are set with the *setWaveField* utility which reads the *waveParameters* file. When using this toolbox the *waveFoam* solver must be used instead of *interFoam*.

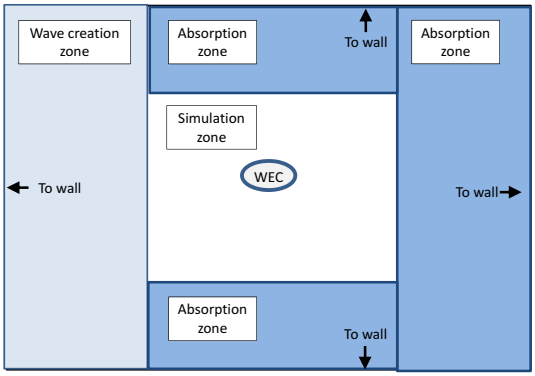

Fig. 14. Relaxation zone configuration for wave creation and absorption

2) Simulating: The solver for the simulation is set in the *controlDict* file in the *system* folder. Other important simulation parameters, such as the simulation length and libraries to be linked, are also set in the *controlDict* file. Selection of the finite-volume techniques for the simulation are set in the *fvSchemes* file. The equation solvers, tolerances and other algorithm controls are set in the *fvSolution* file.

D. Body motion

There are two options available for implementing experiments with body motion; either prescribed or solved motion. Prescribed motion involves updating the body motion along a predefined trajectory, irrespective of the forces acting upon the body. Solved motion involves calculating the body's trajectory from the forces it experiences. The position of the body is a varying quantity throughout the simulation for both options, and the boundary conditions for the body position is set in the *pointDisplacement* file in the *0.org* folder.

1) Prescribed motion: Prescribed motion can be used for radiation type experiments whereby the WEC is oscillated at various frequencies and amplitudes, while the resulting hydrodynamic force on the body measured, as in [17]. Prescribed motion is also used to perform the hydrostatic restoring force experiment in Fig. 10. The present NWT implements prescribed motion using the *oscillatingDisplacement* function. This function sets a sinusoidal device trajectory of variable amplitude and frequency.

2) Solved motion: Used to simulate the motion of rigid-body type WECs in response to input waves and external forces. Implemented by setting the WEC's boundary condition to *calculated* and then calling the *6DoFRigidBodyMotion* solver from the *dynamicMeshDict* file in the *constant* folder.

6DoFRigidBodyMotion calculates the motion using Newton's law; acceleration equals the total force divided by the mass. Here the total force is the hydrodynamic force calculated by the *forces* function plus any additional user defined forces (e.g. PTO and mooring forces). The WEC's mass and moments of inertia are read from the *dynamicMeshDict* file and must be input by the user.

6DoFRigidBodyMotion can constrain the body to only move to designated degrees of freedom via the *constraints* function. Additionally, simple mooring forces can be implemented as linear springs and/or dampers through the *restraints* function. Alternatively, the *restraints* function can be modified to calculate the mooring forces via an external solver [4]. The present NWT implements a PTO force by modifying the *restraints* function, however details of performing the modification are outside the scope of this paper.

E. Parallel Computing

OpenFOAM uses MPI (message passing interface) to allow parallel distribution of the computation across many processors and machines. Domain decomposition is used to break the geometry and associated fields into subdomains which are solved on the separate processors. This process is performed by the *decomposePar* utility, which is controlled by the *decomposeParDict* file in the *system* folder.

F. Recording outputs

This section details the different outputs available from the NWT experiments and how to record them.

1) Body motions: The information regarding position and velocity of the WEC can be found in the solver output log which is written to the *case folder* at run time. At every time-step, the body's centre of mass and velocity are written to this log, along with other information such as the time-step length and current simulation time, number of solver iterations and residuals, execution time, mean and maximum Courant numbers etc. The current NWT then uses *Python* scripts to extract the time, position and velocity values at each time step and store them neatly as vectors in separate text files.

2) Hydrodynamic Forces: The *forces* function integrates the fluid pressure over a surface to give the force from the fluid on that surface. The *libForces.so* library must be first included in the *controlDict* file to use this function. *forces* can then be called from the *controlDict* file to create an output log containing the normal and tangential forces/moments (pressure and shear forces/moments) on the specified surface. The output log is written to a *postProcessing* folder, created at runtime directly inside the *case folder*, and *Python* scripts can be used to extract individual force components, such as heave or pitch, from the output log and store them neatly as vectors in separate text files.

3) Probes: Probes can set inside the *controlDict* file to record pressure and velocity values anywhere in the fluid. The value of these variables is constant in each mesh cell, therefore the spatial resolution of these measurements is defined by the mesh resolution. Wave gauges can be implemented to measure the free surface elevation (FSE) by probing the alpha.water value at many points along a vertical line. An alpha.water value of 1 corresponds to water, 0 to air and the value around

the interface will be somewhere in between 0 and 1. Taking the average of the measured values along the vertical line will tell how far along the line the free surface is located, i.e. if the average alpha.water value probed along a 10m vertical line is 0.735, then the free surface is located 7.35m from the bottom of the line. Alternatively, the *waves2Foam* toolbox has functions which can calculate the FSE which can used by including the *libwaves2Foam.so* library and calling the *surfaceElevation* function in the *controlDict* file.

4) paraView: Paraview is a GUI which allows visualisation of the simulation results, either as snap shots or animations, which is useful for understanding and presenting simulations (see Fig. 15). The *paraFOAM* utility enables paraview to read and display the output results written to the output time directories created during the simulation run. These time directories contain all of the field information at each mesh cell and require a significant amount of disk space to store. The output write interval for the time directories is specified in the *controlDict* file.

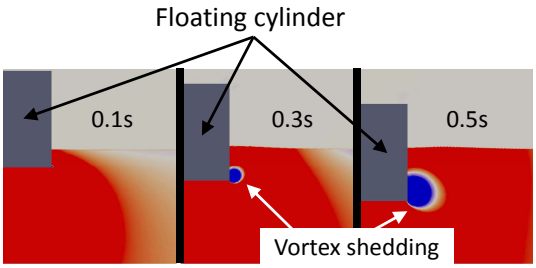

Fig. 15. Paraview postprocess view of the dynamic pressure during a free decay simulation illustrating the creation of vortex structures from the sharp bottom edge of a cylinder during a free decay experiment

G. Debugging

The output logs are full of information from the simulation and are therefore an obvious place to look for debugging purposes. Modifying the C++ source code to write out additional parameters or flags to the output logs can be useful to aide in this purpose.

Using *paraView* to visualise the simulation can offer clues to the cause of a simulation crash. If the output time interval is too sparse, then *paraView* mightn't have information available from the moments preceding the crash and it might be necessary to rerun the simulation with increased temporal resolution for the output files.

The *checkMesh* utility is useful tool for checking the quality of the mesh. *checkMesh* gives the mesh a pass/fail rating as to whether it can be used for simulation and writes an output log reporting on the various mesh quality metrics, such as skewness, aspect ratios etc, and prints the location of bad mesh cells to the *Sets* folder. *checkMesh* should be used after initially creating the mesh to ensure the overall mesh quality before proceeding to simulation and also for debugging purposes after a simulation crash by using *paraView* to display the shape and position of any bad mesh cells output by *checkMesh* to the *Sets* folder.

V. Conclusion

CFD based NWTs are a useful tool for performing wave energy experiments and can be valuable in the design and optimisation of WECs. OpenFOAM is a good choice for implementing a CFD based NWT, due to its open source licensing and growing user base who are constantly verifying and validating its performance and extending and improving its capabilities. Through open sharing by the academic community, on platforms such an OpenORE [46], it may be possible to provide open source NWTs for the wave and other offshore renewable energy industries.

This work in this paper focussed on NWT experiments involving rigid body type WECs and on developing scripts, in the open source software *Python*, to automate the set-up, implementation and post-processing of these experiments. Future work involves implementing this utility for OWC type and flexible body/membrane type WEC experiments.

VI. Acknowledgement

This project is funded by Enterprise Ireland and is co-funded by the Irish Government and the European Union under Irelands EU Structural Funds Programme 2007- 2013 under grant EI/CF/2011/1320.

References

[1] K. Tanizawa, "The state of the art on numerical wave tank," in *Proceeding of 4th Osaka Colloquium on Seakeeping Performace of Ships*, 2000.
[2] I. Currie, *Fundamental Mechanics of Fluids*. McGraw-Hill, 1974.
[3] N. G. Jacobsen, D. R. Fuhrman, and J. Fredsøe, "A wave generation toolbox for the open-source cfd library: Openfoam®," *International Journal for Numerical Methods in Fluids*, vol. 70, pp. 1073–1088, 2012.
[4] J. Palm, C. Eskilsson, G. Moura Paredes, and L. Bergdahl, "Cfd simulation of a moored floating wave energy converter," in *EWTEC 2013*, 2013.
[5] [Online]. Available: https://openfoamwiki.net/index.php/Contrib/waves2Foam
[6] J. Palm. (2012) Connecting openfoam with matlab. Online. Chalmers University of Technology. http://www.tfd.chalmers.se/ hani/kurser/OS CFD 2012/.
[7] *OpenFOAM user guide (available at: www.openfoam.org)*.
[8] H. Nilsson, "Phd course in cfd with opensource software," Available at: `http://www.tfd.chalmers.se/~hani/kurser/OS_CFD/`.
[9] J. Guerrero, "Openfoam course and training," Available at : `http://www.dicat.unige.it/guerrero/openfoam.html`.
[10] G. C. J. Morgan, J. Zang, D. Greaves, A. Heath, C. Whitlow, and J. Young, "Using the rasinterfoam cfd model for wave transformation and coastal modelling," *Coastal Engineering Proceedings*, vol. 1, 2011.
[11] B. Chenari, S. Saadatian, and A. D. Ferreira, "Numerical modelling of regular waves propagation and breaking using waves2foam," *Journal of Clean Energy Technologies*, vol. 3, 2015.
[12] T. Vyzikas, E. Ransley, M. Hann, D. Magagna, D. Greaves, D. Simmonds, V. Magar, and D. Conley, "Integrated numerical modelling system for extreme wave events at the wave hub site," in *ICE Conference on Coasts, Marein Structures and Breakwaters*, 2013.
[13] P. Higuera, J. L. Lara, and I. J. Losada, "Simulating coastal engineering processes with openfoam®," *Coastal Engineering*, vol. 71, pp. 119–134, 2013.
[14] Y. Li and M. Lin, "Wave-body interactions for a surface-piercing body in water of finite depth," *Journal of Hydrodynamics, Ser. B*, vol. 22, 2010.
[15] Y. Li and M. Lin, "Regular and irregular wave impacts on floating body," *Ocean Engineering*, vol. 42, 2012.
[16] L. Chen, J. Zang, A. Hillis, G. Morgan, and A. Plummer, "Numerical investigation of wave–structure interaction using openfoam," *Ocean Engineering*, vol. 88, pp. 91–109, 2014.

[17] L. Bonfiglio, S. Brizzolara, and C. Chryssostomidis, "Added mass and damping of oscillating bodies: a fully viscous numerical approach," *Recent Advances in Fluid Mechanics, Heat & Mass Transfer and Biology*, 2011.

[18] M. Cathelain, "Modelling of a floating object in heave motion using a numerical wave tank in openfoam," Master's thesis, Ecole Centrale de Nantes, 2013.

[19] C. A. Garrido-Mendoza, A. Souto-Iglesias, and K. Thiagarajan, "Numerical simulation of hydrodynamics of a circular disk oscillating near a seabed," in *ASME 2013 32nd International Conference on Ocean, Offshore and Arctic Engineering*, 2013.

[20] J. Gadelho, J. Rodrigues, A. Lavrov, and C. G. Soares, *Maritime Technology and Engineering*. Taylor & Francis Group, London,, 2015, ch. Determining hydrodynamic coefficients of a cylinder with Navier-Stokes equations.

[21] J. Davidson, S. Giorgi, and J. V. Ringwood, "Linear parametric models for ocean wave energy converters identified from numerical wave tank experiments," *Ocean Engineering*, vol. 103, 2015.

[22] J. A. Armesto, R. Guanche, A. Iturrioz, C. Vidal, and I. J. Losada, "Identification of state-space coefficients for oscillating water columns using temporal series," *Ocean Engineering*, vol. 79, pp. 43–49, 2014.

[23] J. Davidson, S. Giorgi, and J. Ringwood, "Numerical wave tank identification of nonlinear discrete-time hydrodynamic models," in *Proc. 1st Int. Conf. on Renewable Energies Offshore (Renew 2014), Lisbon*, 2014.

[24] S. Giorgi, J. Davidson, and J. V. Ringwood, "Identification of nonlinear excitation force kernals using numerical wave tank experiments," in *EWTEC*, 2015.

[25] J. V. Ringwood, J. Davidson, and S. Giorgi, "Optimising numerical wave tanks tests for the parametric identification of wave energy device models," in *Proc. of the 34th Int. Conf. on Ocean, Offshore and Arctic Engineering (OMAE 2015)*, 2015.

[26] J. Palm, C. Eskilsson, L. Bergdahl, and G. Moura Paredes, "Cfd study of a moored floating cylinder: Comparison with experimental data," in *Proc. 1st Int. Conf. on Renewable Energies Offshore (Renew 2014), Lisbon*, 2014.

[27] A. Iturrioz, R. Guanche, J. A. Armesto, C. Vidal, and I. J. Losada, "Experimental and numiercal development of a floating multi-chamber owc device," in *EWTEC 2013*, 2013.

[28] J. Souza, E. dos Santos, and L. Isoldi, "Numerical simulation of an owc devise," in *22nd International Congress of Mechanical Engineering (COBEM 2013), Brazil*, 2013.

[29] P. Schmitt, S. Bourdier, T. Whittaker, D. Sarkar, E. Renzi, F. Dias, K. Doherty, J. van't Hoff *et al.*, "Hydrodynamic loading on a bottom hinged oscillating wave surge converter," in *ISOPE 2012*, 2012.

[30] A. Henry, A. Rafiee, P. Schmitt, F. Dias, and T. Whittaker, "The characteristics of wave impacts on an oscillating wave surge converter," in *Proceedings of the 23rd International Offshore and Polar Engineering Conference*, 2013.

[31] H. Akimoto, Y. Kim, and K. Tanaka, "Performance prediction of the rotational wave energy converter using single-bucket drag type turbine," in *ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering*, 2014.

[32] J. Thaker and J. Banerjee, "Numerical simulation of periodic motion of two phases in a two-dimensional channel: application to wave energy converter," in *Proceedings of the 39th National Conference on Fluid Mechanics and Fluid Power, India*, 2012.

[33] A. King, "Numerical modelling of the bomborawave energy conversion device," in *19th Australasian Fluid Mechanics Conference, Melbourne, Australia*, 2014.

[34] C. Eskilsson, J. Palm, J. P. Kofoed, and E. Friis-Madsen, "Cfd study of the overtopping discharge of the wave dragon wave energy converter," in *Proc. 1st Int. Conf. on Renewable Energies Offshore (Renew 2014), Lisbon*, 2014.

[35] M. A. Afshar, "Numerical wave generation in open foam," Master's thesis, Chalmers University of Technology, 2010.

[36] J. Alsgaard, "Numerical investigations of piston mode resonance in a moonpool using openfoam," Master's thesis, Norwegian University of Science and Technology, 2010.

[37] J. Andersson, "Simulation of wave induced forces on semi submerged horizontal cylinders using openfoam," Master's thesis, Chalmers University of Technology, 2011.

[38] R. J. Lambert, "Development of a numerical wave tank using openfoam," Master's thesis, Universidade De Coimbra, 2012.

[39] B. T. Paulsen, "Efficient computations of wave loads on offshore structures," Ph.D. dissertation, Technical University of Denmark, 2013.

[40] G. C. Morgan, "Application of the interfoam vof code to coastal wave/structure interaction," Ph.D. dissertation, University of Bath, 2013.

[41] J. Palm, "Developing computational methods for moored floating wave energy devices," Ph.D. dissertation, Department of Shipping and Marine Technology, Chalmers University of Technology, 2014.

[42] P. Q. Thien, N. K. Hieu, and P. M. Vuong, "Numerical simulation of floating airboat: Estimation of hydrodynamic forces," *Int. J.of Mech. Eng. and Applications*, 2015.

[43] M. A. Benitz, D. P. Schmidt, M. A. Lackner, G. M. Stewart, J. Jonkman, and A. Robertson, "Comparison of hydrodynamic load predictions between reduced order engineering models and computational fluid dynamics for the oc4-deepcwind semi-submersible," in *ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering*, 2014.

[44] S. Dai, B. A. Younis, and L. Sun, "Openfoam predictions of hydrodynamics loads on full-scale TLP," *Ocean Engineering*, vol. 102, 2015.

[45] L. Ding, L. Zhang, C. Wu, X. Mao, and D. Jiang, "Flow induced motion and energy harvesting of bluff bodies with different cross sections," *Energy Conversion and Management*, vol. 91, 2015.

[46] [Online]. Available: www.openore.org