waves2Foam Manual

Technical Report · August 2017

CITATIONS
0

READS
4,985

1 author:

Niels Gjøl Jacobsen
Deltares
30 PUBLICATIONS **648** CITATIONS

SEE PROFILE

Niels G. Jacobsen

waves2Foam Manual

August 2017, version 0.9 (SVN-revision 2113)
DRAFT

WAVES2FOAM MANUAL

Niels G. Jacobsen

Front matter: Own picture (Ireland, 2016)
The manual is published digitally

Release history:

Date	Version	Latest SVN-revision	Affiliation
August 2017	0.9 (DRAFT)	2113	Deltares, The Netherlands (niels.jacobsen@deltares.nl)

Contents

Code Frames

Nota Benes

Warnings

CHAPTER 1

Introduction

1.1 Origin

The waves2Foam toolbox was originally developed at the Technical University of Denmark by Niels Gjøl Jacobsen under the supervision of Prof. Jørgen Fredsøe. The porosity module was developed in collaboration between Bjarne Jensen and Niels Gjøl Jacobsen, both then at the Technical University of Denmark.

As of the employment of Niels G. Jacobsen at Deltares, The Netherlands (2013-present), the maintenance and further developments takes place at Deltares.

1.2 Background and application

The waves2Foam toolbox is a plug-in to the open-source general purpose CFD-package OpenFoam®[1,2]. waves2Foam is available as open-source under the GNU General Public License (GNU GPL).

waves2Foam was initially only released as a toolbox for the generation and absorption of free surface waves (Jacobsen et al., 2012) in November 2011. In 2014, the toolbox was extended with the possibility of modelling the interaction between free-surface waves and a permeable medium such as breakwaters, scour protection, etc (Jensen et al., 2014).

The waves2Foam toolbox is used extensively in a large number of references and it ranges from validation to application of the toolbox and the free surface capabilities of OpenFoam. Examples of the usage of waves2Foam is given here:

- Jacobsen et al. (2014); Jacobsen and Fredsøe (2014a,b) applied the model to study the evolution of cross-shore morphological development under breaking waves. This involved a coupling between the hydrodynamics, sediment transport and the resulting bed level change.
- Paulsen et al. (2014a,b) used the toolbox to study wave loads on a monopile. The work by Paulsen et al. (2014a) furthermore included a coupling between waves2Foam and a solution to the full non-linear wave problem, i.e. the Laplace equation with non-linear boundary conditions; more details are given in Section 4.3.3.
- The resonance of the surface elevation within moon pools (Moradi, 2015).
- The study of wave loads on bridge decks, where comparions with experimental work were presented (Seiffert, 2014; Seiffert et al., 2014; Hayatdavoodi et al., 2014).
- Jacobsen et al. (2015) validated waves2Foam against numerous laboratory experiments with normal incident irregular waves. This showed that the relaxation zone approach for the generation and absorption in waves2Foam compensates for the additional mass flux due to the Stokes drift without a increase in the water level.
- Stahlmann (2013) coupled the wave generation in waves2Foam with a method for the deformation of the sediment bed around a tripod steel structure; the purpose was to evaluate the effects of this structure on scour patterns.

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

[2] OPENFOAM® is a registered trade mark of OpenCFD Limited, producer and distributor of the OpenFOAM software via www.openfoam.com.

- Elsafti and Oumeraci (2013) coupled waves2Foam with a geotechnical model to study the residual pore pressure build-up and dissipation underneath a caisson breakwater.

The list of application is steadily expanding, why the most up-to-date information is to look at the citing papers on e.g. Google Scholar (https://scholar.google.com) or ResearchGate (www.researchgate.net).

1.3 Compabitility

1.3.1 Branches

OpenFoam comes in three main branches:

- OpenFoam (OF) as distributed through www.openfoam.org.
- OpenFoam+ (OF+) as distributed through www.openfoam.com.
- foam-extend (FE) as distributed through the foam-extend community.

The abbreviation in the parenteses are used throughout this document.

1.3.2 Naming terminology

waves2Foam is supported on all three branches of OpenFoam and on a long list of version numbers. Over the years, the various branches have modified the naming terminology. For instance, the void fraction ratio, F, is termed `alpha1` in some versions and `alpha.water` in other versions. Similarly, the excess pressure is termed `pd` in some versions and `p_rgh` in other versions. These differences in naming terminology are supported by waves2Foam through the class `<waves2Foam_src>/waves2Foam/include/crossVersionCompatibility.[C,H]`.

1.3.3 Supported versions

waves2Foam compiles on the following (recent) versions, however, please refer to Warning 3.1:

- OF-3.0, OF-4.0
- OF+-3.0, OF+-1606, OF+-1612
- FE-3.1, FE-3.2, FE-4.0

> **Warning 1.1:** Numerical wave gauges does not compile on recent OpenFoam versions (OF and OF+).
>
> The surface sampling utility (numerical wave gauges) is a vital part of modelling free surface waves with OpenFoam. Due to a code reorganisation, the wave gauges do not work for OF-4.0 (and newer) and OF+-1612 (and newer).
>
> This issue has to be resolved in a future update of waves2Foam.

1.4 Installation of waves2Foam

waves2Foam is straightforward to install, if the necessary third-party packages are already available on the computer. This section details the installation procedure.

1.4.1 Third-party dependencies

The following third-party dependencies are required to install waves2Foam. These dependencies are needed in addition to those required to compile OpenFoam itself.

- `gfortran` is needed to compile OceanWave3D and Fenton's code for the stream function coefficients.
- `git` is required to access the source code for OceanWave3D.
- `subversion` (SVN) is required to access the source code for waves2Foam.
- GNU Scientific Library (GSL) is required to compile waves2Foam. GSL provides various mathematical functions not natively supported in OpenFoam.

Please refer to the package management system on your flavour of Linux/UNIX for the installation of these third-party packages. Compilation of waves2Foam is unsuccessful, if any of these packages are missing.

1.4.2 Download and installation

The download and installation instructions are for a default installation location (Code Fragment 1.1). If the installation is in a non-default location, please refer to Section 1.4.3 for additional actions before `$ ./Allwmake`.

Code Fragment 1.1: Download and installation instructions for the default installation directory (`$` refer to a command).

```
## First, source your OpenFoam version
$ mkdir -p $FOAM_RUN/../applications/utilities
$ cd $FOAM_RUN/../applications/utilities
svn co http://svn.code.sf.net/p/openfoam-extend/svn/trunk/\
    Breeder_1.6/other/waves2Foam
$ cd waves2Foam
$ ./Allwmake
```

1.4.3 The `bashrc` file

waves2Foam is distributed with a control file called `bashrc.org`, which is located in the directory `bin`. In case of a default installation, this file is automatically copied to `bin/bashrc`. The file controls the linking to GNU Scientific Library and the target directories for libraries and executables.

If a non-standard installation is required, the following set of user-defined variables may be modified:

- `WAVES_DIR`: This is the full path to the waves2Foam-installation.
- `WAVES_APPBIN`: This is the location for the executables. The default value is the user-directory (`$FOAM_USER_APPBIN`), though an alternative is `$FOAM_APPBIN`, which is useful for cluster installations with multiple users.
- `WAVES_LIBBIN`: This is the location for the libraries. The default value is the user-directory (`$FOAM_USER_LIBBIN`), though an alternative is `$FOAM_LIBBIN`, which is useful for cluster installations with multiple users.
- `WAVES_GSL_INCLUDE`: Header files for GNU Scientific library (`gsl` is appended during compilation).
- `WAVES_GSL_LIB`: Path to the GNU Scientific libraries.

It is recommended never to modify `bashrc.org`. Always make a copy and make modifications in this file. Modifications to `bashrc.org` could cause problems, when updating the SVN-repository.

1.4.4 How to get the updates to waves2Foam?

Updates to waves2Foam are obtained by updating the SVN-repository, followed by recompilation. This is outlined in Code Fragment 1.2.

Code Fragment 1.2: Updating the SVN-repository and recompilation of waves2Foam.

```
$ svn update
$ ./Allwmake
```

Updates to the SVN-repository are normally announced through CFD-Online in the thread called "Release of a Wave Generation and Absorption Toolbox for OF". The easiest is to subscribe to the thread and receives email-notifications.

All revisions of waves2Foam are detailed in Appendix C.

Nota Bene 1.1: `bashrc after updates`

Please note that the file `bashrc.org` might get updated with new versions of waves2Foam. If this happens, it is recommended to use the most up-to-date version of `bashrc.org`, see instructions in Section 1.4.3.

1.4.5 How to recompile parts of waves2Foam

Sometimes it is needed to recompile parts of the code, if modifications or additions have been made. It is *not* possible to only recompile parts of the code, due to the temporary definition of environmental variables by the inclusion of `bin/bashrc`. Therefore, waves2Foam should always be compiled using the `Allwmake` script.

1.5 Referencing

The users are reguested to support the work put into the waves2Foam toolbox by providing proper referencing to the academic work that has been supporting the various releases:

Any use of waves2Foam: Please make reference to the original work (Jacobsen et al., 2012) as specified in Code Fragment 1.3.

Code Fragment 1.3: Reference to the original paper accompanying the release of waves2Foam.

```
@article{jacobsenFuhrmanFredsoe2012,
Author = {Jacobsen, N G and Fuhrman, D R and Freds\o{}e, J},
Title = {{A Wave Generation Toolbox for the Open-Source CFD Library:
    OpenFoam\textregistered}},
Journal = {{International Journal for Numerical Methods in Fluids}},
Year = {{2012}},
Volume = {{70}},
Number = {{9}},
Pages = {{1073-1088}},
DOI = {{10.1002/fld.2726}},
}
```

Use of the porosity module: Please make reference to the derivation of the Navier-Stokes equations in terms of filter velocities for permeable structures (Jensen et al., 2014) as specified in Code Fragment 1.4.

Code Fragment 1.4: Reference to the paper on the porosity implementation in waves2Foam.

```
@article{jensenJacobsenChristensen2014,
Author = {Jensen, B and Jacobsen, N G and Christensen, E D},
Title = {{Investigations on the porous media equations and resistance
    coefficients for coastal structures}},
Journal = {{Coastal Engineering}},
Year = {{2014}},
Volume = {{84}},
Pages = {{56-72}},
}
```

Coupling with OceanWave3D: When the coupling between waves2Foam and OceanWave3D is used, please make reference to Paulsen et al. (2014a) for the coupling (Code Fragment 1.5) and reference to Engsig-Karup et al. (2009) for the development of OceanWave3D (Code Fragment 1.6).

The coupling was later streamlined into the waves2Foam framework and it fits into the framework as an external source, see Section 4.2.8).

Code Fragment 1.5: Reference to the coupling between waves2Foam and OceanWave3D.

```
@article{ paulsenBredmoseBingham2014,
  Author = {Paulsen, B. T. and Bredmose, H. and Bingham, H. B.},
  Title = {{An efficient domain decomposition strategy for wave loads on
       surface piercing circular cylinders}},
  Journal = {{Coastal Engineering}},
  Year = {{2014}},
  Volume = {{86}},
  Pages = {{57-76}},
}
```

Code Fragment 1.6: Reference to OceanWave3D.

```
@article{ engsigKarupBinghamLindberg2009,
Author = {Engsig-Karup, A. P. and Bingham, H. B. and Lindberg, O.},
Title = {{An efficient flexible-order model for 3D nonlinear water waves}},
Journal = {{Journal of Computational Physics}},
Year = {{2009}},
Volume = {{228}},
Number = {{6}},
Pages = {{2100-2118}},
DOI = {{10.1016/j.jcp.2008.11.028}},
}
```

CHAPTER 2

Mathematical Description

2.1 Coordinate system

The coordinate system is defined by the direction of the gravity vector given in the file `constant/g`. This means that the vertical coordinate can be any direction; theoretically. Only definition of the gravity vector along the y- or z-axis is extensively tested.

2.2 Momentum and continuity equations

The momentum and continuity equations follow the standard implementation in OpenFoam for Volume of Fluid (VOF) type simulations. The momentum equation is written in the excess pressure that is defined as the pressure in excess of the hydrostatic pressure. The momentum equation reads:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot \rho \mathbf{u}\mathbf{u}^T = -\nabla p^* + \mathbf{g} \cdot (\mathbf{x} - \mathbf{x}_r)\nabla \rho + \nabla \cdot \mu_{tot} \nabla \mathbf{u} \tag{2.1}$$

Here, ρ is the density of the fluid, $\mathbf{u}$ is the velocity vector in Cartesian coordinates, t is time, $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$ is the differential operator, p^* is the excess pressure, $\mathbf{g}$ is the vector of acceleration due to gravity, $\mathbf{x} = (x, y, z)$ is the Cartesian coordinate vector, $\mathbf{x}_r$ is a reference location (defined at sea level, see Section 3.1.2) and μ_{tot} is the total dynamic viscosity, see Section 2.4.

The incompressible continuity equation reads:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.2}$$

The excess pressure is defined as $p^* = p - \rho \mathbf{g} \cdot \mathbf{x}$, where p is the total pressure. The solution procedure for the discretised equations are outlined in Rusche (2002).

2.2.1 Modifications for permeable structures

In the case of permeable structure, such as breakwaters, scour protection, a permeable sea bottom, etc, the velocity is defined as the filter velocity. The filter velocity is related to the pore velocity through the porosity, n, as follows:

$$\mathbf{u} = n\mathbf{u}_p \tag{2.3}$$

$\mathbf{u}_p$ is the pore velocity. The derivation of the implementation of the correction due to permeable layers is presented in Jensen et al. (2014). The continuity equation is the same as in Eq. (2.2), while the momentum equation takes the following modified version:

$$(1 + C_m)\frac{\partial}{\partial t}\frac{\rho \mathbf{u}}{n} + \frac{1}{n}\nabla \cdot \frac{\rho}{n}\mathbf{u}\mathbf{u}^T = -\nabla p^* + \mathbf{g} \cdot (\mathbf{x} - \mathbf{x}_r)\nabla \rho + \frac{1}{n}\nabla \cdot \mu_{tot}\nabla \mathbf{u} - \mathbf{F}_p \tag{2.4}$$

This form of the momentum equation deviates from the one presented in Higuera et al. (2014), however, in a later work the main author (Higuera, 2015) aligned himself with the formulation in Eq. (2.4). The quantities C_m and $\mathbf{F}_p$ are discussed in Section 2.8. The continuity equation is still given by Eq. (2.2).

This modelling framework has successfully been applied to study the interaction between waves and permeable structure (Jensen et al., 2014; Jacobsen et al., 2015; Van Gent et al., 2015). These works have validated quantities such as reflection, wave dissipation, wave overtopping and wave induced pressures and setup inside permeable structures.

2.3 Advection of the indicator field

The advection of the VOF-field, F, is done as follows:

$$\frac{\partial F}{\partial t} + \nabla \cdot \mathbf{u}F + \nabla \cdot \mathbf{u}_r(1-F)F = 0 \tag{2.5}$$

Here, $\mathbf{u}_r$ is a relevative velocity, see e.g. Berberović et al. (2009) for details. F is used to evaluate the spatial variation in density and viscosity as follows:

$$\rho = F\rho_1 + (1-F)\rho_0 \quad \text{and} \quad \mu = F\mu_1 + (1-F)\mu_0 \tag{2.6}$$

Subscript 0 refers to material properties corresponding to $F = 0$ and subscript 1 refers to material properties corresponding to $F = 1$. It is assumed that $F = 1$ for water and $F = 0$ for air in waves2Foam.

2.3.1 Modifications for permeable structures

The presence of a permeable structure means that the amount of fluid in a cell is limited to the voids between the permeable material. It is still desirable to have $F \in [0, 1]$, why the advection equation effectively takes this work:

$$\frac{\partial F}{\partial t} + \frac{1}{n}\left(\nabla \cdot \mathbf{u}F + \nabla \cdot \mathbf{u}_r(1-F)F\right) = 0 \tag{2.7}$$

n is the porosity.

2.4 Turbulence modelling

No turbulence models are distributed with waves2Foam. There are plenty of turbulence models available in the various distributions of OpenFoam, however, please be aware of recommendations given in the literature (some examples are Mayer and Madsen, 2000; Jacobsen et al., 2012; Brown et al., 2016; Zhou et al., 2017; Devolder et al., 2017)

2.5 Reflection compensation at the boundary

Techniques are known to perform reflection compensation directly at the boundary (Wellens, 2012; Higuera, 2015), though such methods are not available for waves2Foam, which is developed around the use of relaxation zones.

2.6 Numerical beach

waves2Foam is prepared for the implementation of a numerical beach. This implementation has never been completed. Some tutorials contains a keyword `beachType` in the defintion of the relaxation zone, though a default value is assigned, why this keyword is not required.

2.7 Relaxation zone technique

The relaxation zone technique is meant to remove spurious reflection from numerical simulations. The relaxation zone technique is based on a weighting between the computed solution of the velocity field and the indicator field with a target solution. The relaxation zone technique is divided into two flavours, namely explicit and implicit relaxation, where explicit/implicit refers to the time integration.

Furthermore, the specification of the weighting and location of the relaxation zones in the computational domain are described.

2.7.1 Flavour of relaxation

2.7.1.1 Explicit relaxation

The explicit approach in the relaxation zones is simply given as:

$$\phi = (1 - w_R)\phi_{\text{target}} + w_R\phi_{\text{computed}} \tag{2.8}$$

Here, the weighting function $w_R \in [0, 1]$ can be defined in various ways as will be described in the following sections.

This method explicitly corrects the fields α and $\mathbf{u}$ according to Eq. (2.8) each time step prior to the solution to the pressure-velocity coupling. The method was originally described in Jacobsen et al. (2012) for OpenFoam, though it is a fairly common approach for e.g. Boussinesq type wave modelling or solution to the full Laplace problem (see e.g. Fuhrman et al., 2006; Engsig-Karup et al., 2009).

2.7.1.2 Implicit relaxation

An implicit relaxation zone implementation has been proposed by Vukcević et al. (2016a,b), though the method is currently not available in waves2Foam.

2.7.2 Specifying the relaxation zone

The relaxation zone is specified in the file `waveProperties.input` inside a sub-dictionary, see Section 3.1.2 for more details. The relaxation zone is defined as in Code Fragment 2.1. Each of these controls are described individually in the following sections.

Code Fragment 2.1: Fundamental definition of a relaxation zone

```
relaxationZone
{
    relaxationScheme    <word>;
    relaxationWeight    <word>;
    relaxationShape     <word>;
    courantCorrection   <bool>; // Default: false
}
```

2.7.2.1 Relaxation scheme

Relaxation scheme is essentially the way, the relaxation is implemented.

relaxationSchemeEmpty This method effectively turns of the relaxation (Code Fragment 2.2).

Code Fragment 2.2: Empty relaxation scheme

```
relaxationZone
{
    relaxationScheme    Empty;
}
```

relaxationSchemeSpatial The spatial relaxation scheme performs the relaxation according to Eq. (2.8) (Code Fragment 2.3).

Code Fragment 2.3: Spatial relaxation scheme

```
relaxationZone
{
    relaxationScheme    Empty;
}
```

2.7.2.2 Relaxation weight

There are three relaxation weights available and each of these can be corrected for the global Courant number. The following weights are available:

- Exponential weight (default)
- Free polynomial weight
- Third order polynomial weight

These methods produce a weight, w_R, which can be corrected based on the local Courant number as follows:

$$\tilde{w}_R = 1 - (1 - w_R^*)^{\mathrm{Co}/\mathrm{Co}_{\max}} \tag{2.9}$$

where $w_R^* = 1 - w_R$ and $\tilde{w}_R$ is used in Eq. (2.8). Here, Co is the local Courant number and $\mathrm{Co}_{\max}$ is the maximum Courant number. The method is implemented following the work by Seng (2012). The effect is activated by adding the keyword `courantCorrection` in the definition of the relaxation zone, see Code Fragment 2.1.

In all cases, the weights are a function of a local coordinate system in the relaxation zone, such that $w_R(\sigma = 1) = 0$ and $w_R(\sigma = 0) = 1$. Here, σ is a local coordinate within the relaxation zone, where the coordinate depends on the shape of the relaxation zone.

Exponential weight The exponential weight distribution is taken from Fuhrman et al. (2006) and it has the following form:

$$w_R = 1 - \frac{\exp \sigma^p - 1}{\exp 1 - 1} \tag{2.10}$$

The exponent p is set to 3.5 as default. The method is default and there is no need to specify it, but it can be explicitly included as in Code Fragment 2.4:

Code Fragment 2.4: Exponential weight function

```
relaxationWeight    Exponential;   // Default
exponent            <scalar>;      // Default value = 3.5
```

Free polynomial weight The free polynomial weight is encountered in Engsig-Karup (2006) and the weight function is given as:

$$w_R = 1 - \sigma^p \tag{2.11}$$

where p is an integer exponent. The weight can be specified as in Code Fragment 2.5. There is no default value for the exponent p.

Code Fragment 2.5: Free polynomial weight function

```
relaxationWeight    FreePolynomial;
exponent            <scalar>;
```

Third-order polynomial weight The third order polynomial weight is encountered in Engsig-Karup (2006) and the weight function is given as:

$$w_R = -2\tilde{\sigma}^3 + 3\tilde{\sigma}^2 \tag{2.12}$$

where $\tilde{\sigma} = 1 - \sigma$. The weight can be specified as in Code Fragment 2.6

Code Fragment 2.6: Third order polynomial weight function

```
relaxationWeight    ThirdOrderPolynomial;
```

2.7.2.3 Relaxation shape

The relaxation shape specifies, where in the computational domain a certain relaxation zone is applied. There are currently four different relaxation shapes available. These are:

- Rectangular;
- Semi-cylindrical;
- Cylindrical;

- Frozen

All of the relaxation zones update the affected cell indices every time step, if the mesh is changing (moving or mesh refinement). The exception to this is the frozen relaxation zone, see below. The settings for each of these are as follows. For all case, the vertical axis is defined based on the direction of gravity.

Rectangular The rectangular shape is the most useful shape, because it works equally well for 2D and 3D simulations. It is defined based on the coordinates of the two diagonal corner points of a rectangle and the direction of one of the lateral sides (and direction of relaxation). The formatting of the relaxation shape is given in Code Fragment 2.7

Code Fragment 2.7: Rectangular relaxation shape

```
relaxationShape     Rectangular;
relaxType           <word>;
startX              <point>;
endX                <point>;
orientation         <vector>;
```

The keyword `relaxType` takes either of two words, namely `INLET` and `OUTLET`. The former is used to defined that the direction of the vector `orientation` describes the direction of relaxation, whereas the use of `OUTLET` means that the direction of relaxation is *opposite* to that prescribed by `orientation`, see more in Section 3.1.3.

All points must be in the same horizontal plane.

Semi-cylindrical A semi-cylindrical shape can be defined based on the specifications in Code Fragment 2.8. This method is perhaps not particularly useful, but it shows an example of how alternative shapes may be defined.

Code Fragment 2.8: Semi-cylindrical relaxation shape

```
relaxationShape     SemiCylindrical;
centre              <point>;
rInner              <scalar>;
rOuter              <scalar>;
zeroAngleDirection  <vector>;
angleStart          <scalar>;    // [degrees]
angleEnd            <scalar>;    // [degrees]
```

Cylindrical The cylindrical relaxation shape defined a relaxation zone with an inner and outer radious (Code Fragment 2.9). The target solution is enforced at the outer edge of the cylindrical relaxation zone. Until now, the only usage of this relaxation zone appears to be the work by Arrighi et al. (2015).

Code Fragment 2.9: Cylindrical relaxation shape

```
relaxationShape     Cylindrical;
centre              <point>;
rInner              <scalar>;
rOuter              <scalar>;
```

Frozen The frozen relaxation shape is special in the way that is mimics any of the other relaxation shapes, except for one aspect: *It does not update the cell indices for the relaxation zone, if the computational mesh is moving.* The method is not applicable on meshes with topological changes. The relaxation shape is defined as given in Code Fragment 2.10

Code Fragment 2.10: Frozen relaxation shape

```
relaxationShape       Frozen;
actualRelaxationShape <word>;

// Additional lines for the actual relaxation shape
```

2.8 Specification of resistance due to a permeable structure

The resistance force in waves2Foam is of the form (Eq. (2.4)):

$$\frac{\mathbf{F}_p}{\rho} = a\mathbf{u} + b\mathbf{u}\|\mathbf{u}\|_2 \tag{2.13}$$

where a and b are resistance coefficients. In additional to this, the added mass coefficient C_m is given as follows:

$$C_m = \gamma_p \frac{1-n}{n} \tag{2.14}$$

Here, γ_p is a closure coefficient is taken as 0.34.

2.8.1 Resistance formulations

There are three resistance formulations implemented in waves2Foam. The required coefficients for each of these are detailed in the following subsections.

2.8.1.1 Native OpenFoam

The native formulation reads two parameters `d` and `f` (Code Fragment 2.11).

Code Fragment 2.11: The native relaxation zone

```
d  d [0 -2 0 0 0 0 0]        <vector>;
f  f [0 -1 0 0 0 0 0]        <vector>;
porosity                     <scalar>;
gammaAddedMass               <scalar>;
```

2.8.1.2 Engelund (1953)

The resistance coefficients a and b take the following form in the formulation by Engelund (1953):

$$a = \alpha \frac{(1-n)^3}{n^2} \frac{\nu}{d_{50}^2} \quad , \quad b = \beta \frac{1-n}{n^3} \frac{1}{d_{50}} \tag{2.15}$$

Here, ν is the kinematic viscosity and d_{50} is the nominal, median diameter of the grain material. The input parameters are specified in Code Fragment 2.12.

Code Fragment 2.12: Resistance formulation according to Engelund (1953)

```
resistanceFormulation        engelund1953;

porosity                     <scalar>;
gammaAddedMass               <scalar>;

d50   d50   [0 1 0 0 0 0 0]  <scalar>;
alpha alpha [0 0 0 0 0 0 0]  <scalar>;
beta  beta  [0 0 0 0 0 0 0]  <scalar>;
```

2.8.1.3 Van Gent (1995)

The resistance coefficients a and b take the following form in the formulation by Van Gent (1995):

$$a = \alpha \frac{(1-n)^2}{n^3} \frac{\nu}{d_{50}^2} \quad , \quad b = \beta \left(1 + \frac{7.5}{KC}\right) \frac{1-n}{n^3} \frac{1}{d_{50}} \tag{2.16}$$

Here, ν is the kinematic viscosity and d_{50} is the nominal diameter of the grain material. The input parameters are specified in Code Fragment 2.13.

Code Fragment 2.13: Resistance formulation according to Van Gent (1995)

```
resistanceFormulation        vanGent1995;

porosity                     <scalar>;
gammaAddedMass               <scalar>;

d50   d50   [0 1 0 0 0 0 0]  <scalar>;
alpha alpha [0 0 0 0 0 0 0]  <scalar>;
beta  beta  [0 0 0 0 0 0 0]  <scalar>;
KC    KC    [0 0 0 0 0 0 0]  <scalar>; // Default: 10000
```

The inserted values are merely examples. For choice of α, β and KC, see e.g. Jensen et al. (2014); Jacobsen et al. (2015); Losada et al. (2016). Furthermore, please note that the effect of the KC number is being discussed in Jacobsen et al. (In preparation).

CHAPTER 3

Utilities

This chapter discusses the range of pre- and post-processing utilities.

3.1 Pre-processing utilities

3.1.1 waveGaugesNProbes

The utility allows for the definition of probes and wave gauges. The definition is outlined in the following.

3.1.1.1 Input:

The file `constant/probeDefinitions` is used to specify the location of the wave gauges and probes. The common settings are defined in Code Fragment 3.1.

Code Fragment 3.1: Common settings for definition of wave gauges and probes

```
<name of wave gauges>
{
    type waveGauge;           // Definition of the type

    add       (0 0 1);        // Length and orientation of gauge
    axis      z;              // Vertical axis

    // Insert point distribution
}

<name of probes>
{
    type probeGauge;

    outputInterval <label>;   // The frequency of outputting
    fields (<name0> <name1>); // The fields to sample

    // Insert point distribution
}
```

The point distributions to be provided (Code Fragment 3.1) are defined in the following.

Warning 3.1: Definition of wave gauges

All wave gauges must be defined, such that they are at least partly inside of the computational domain. No checks are made on the location of the wave gauges relative to the computational domain and the computation will crash without any explanation, if one or more wave gauges are completely outside of the computational domain.

circularDistribution Defines a circular distribution with a certain radius around a given centre. The point distribution is defined as in Code Fragment 3.2.

Code Fragment 3.2: A circular point distribution

```
pointDistribution circularDistribution;
N                 4;
centre            (0 0 0);
radius            1;
```

lineDistribution Defines a linear distribution along a line defined by a starting and an end point. It is possible to have the distribution stretched along the line; resulting in a non-equidistant point distribution (Code Fragment 3.3).

Code Fragment 3.3: A linear point distribution

```
pointDistribution lineDitribution;
N                 4;
linestart         (0 0 0);
lineend           (1 0 0);
stretch           1.;
```

quadrilateralDistribution Defines a quadrilateral point distribution, i.e. it is possible to define a rectangular distribution of the wave gauges with e.g. $3 \times 3 = 9$ wave gauges (Code Fragment 3.4).

Code Fragment 3.4: A quadrilaterial point distribution

```
pointDistribution quadrilateralDistribution;
N0                10;
N1                11;
linestart0        (0 0 0);
lineend0          (1 0 0);
lineend1          (1 1 0);
stretch0          0.9;       // Default = 1.0
stretch1          1.1;       // Default = 1.0
```

userDefinedDistribution The user defined distribution is conveniently used, when the wave gauges should be distributed according to locations in a physical experiment (Code Fragment 3.5).

Code Fragment 3.5: User-defined point distribution

```
pointDistribution userDefinedDistribution;
N       4;
xValues nonuniform List<scalar> 4(0 1 2 3);
yValues nonuniform List<scalar> 4(0.1 0.2 0.3 0.4);
zValues uniform 0;
```

All of the locations (`xValues`, `yValues` and `zValues`) can be defined according to either `nonuniform` or `uniform` formatting.

3.1.1.2 Output:

There are two (probe gauges) or four (wave gauges) outputs from `waveGaugesNProbes` per sub-dictionary in `constant/probeDefinitions`. All output are written to the folder 'waveGauges-NProbes' in the case folder. The files are detailed here:

- `<name of set>_controlDict`: This file contains the part that should be copied to the controlDict for run-time sampling with the wave gauges, see e.g. Section 3.1.1.3 for a description.

- `<name of set>_sets` (only wave gauges): This file contains the definition of the wave gauges, which is essentially the definition of vertical lines in the standard format for the 'sample' utility in OpenFoam.

- `<name of set>surfaceElevationDict` (only wave gauges): This file is to be used for sampling of the wave gauges as a post-processing step. Simply copy the file to the system folder and execute the command `surfaceElevation` (Section 3.3.1).

- `<name of set>.vtk`: This file can be opened with ParaView/paraFoam in order to visualise the the location of the probe and wave gauges.

3.1.1.3 Run-time sampling

The run-time sampling is exemplified with the wave gauges. Similar approach holds for the probe gauges.

The surface elevation can be sampled with the pre-defined wave gauges as either a post-processing step or as a run-time evaluation. The latter is discussed here and the former is described in Section 3.3.1. A part of the output from `waveGaugesNProbes` is a file called `<name of set>_controlDict` (Code Fragment 3.6)

Code Fragment 3.6: Example of the control of the surface elevation sampling.

```
surfaceElevation
{
    type               surfaceElevation;
    functionObjectLibs ( "libwaves2Foam.so" );

    outputControl      timeStep; // Alternative: outputTime
    outputInterval     1;

    //Additional output controls in waves2Foam
    //samplingStartTime  -1;
    //surfaceSampleDeltaT 0.025;

    setFormat          raw;
    interpolationScheme cellPointFace;
    fields (alpha1);

    #includeIfPresent "../waveGaugesNProbes/surfaceElevationAnyName_sets";
}
```

Note that the name of the indicator field `alpha1` is automatically adjusted to the default naming a given version of OpenFoam.

The ordinary outputControls work for this utility, but it is also possible to sample the surface elevation at an approximately equidistant time step, which is much smaller than the output time of the fields (see commented lines in Code Fragment 3.6). Note, irrespectively of the value of `surfaceSampleDeltaT`, the sampling utility is only visited based on the standard OpenFoam controls (`outputControl` and `outputInterval`). It is recommended to use the default settings.

3.1.2 setWaveParameters

The utility `setWaveParameters` is a pre-processing utility, which computes all the necessary wave parameters based on physical meaningful properties, e.g. `setWaveParametes` converts information on water depth and wave period into a wave number for first order Stokes wave theory.

All input is written to the file `<casePath>/constant/waveProperties.input` and the processed data is outputted in `<casePath>/constant/waveProperties`. This sub-division into two files comes from historical reasons.

The input file `waveProperties.input` contains global information and information related to each wave boundary and/or relaxation zone. The global parameters and the setup for a boundary condition and/or relaxation zone is given in the following two sections.

Code Fragment 3.7: Globale settings in `waveProperties.input`

```
// Only relevant for porousWaveFoam
porosityModel          jensenJacobsenChristensen2014;

// Global definition of still water level
seaLevel               <scalar>;
sealevelAsReference    <boolean>;

// Names of relaxation zones. The wave theories are defined in the
// subdictionaries name0Coeffs, name1Coeffs, etc.
relaxationNames        (name0 name1 ...);

// Wave theory used to initiate the simulation.
initializationName     outlet;

// Not necessary. Default value is 'emptyExternal'.
externalForcing        emptyExternal;
```

3.1.2.1 Global parameters

The global settings are specified in Code Fragment 3.1.2.1. The number of relaxation zones are defined through the list `relaxationNames`. It also means that a relaxation zone can be defined in the sub-dictionaries, e.g. `name0Coeffs`, but if the string `name0` is not written in the list `relaxationNames`, then the relaxation zone is not active.

The external forcing is an option to provide the simulation with external wave data, see Section 4.3 for adescription of the options.

3.1.2.2 Sub-dictionary information

Each set of wave properties are given in a sub-dictionary, which is called `<name> + "Coeffs"`. An example of the structure of such a sub-dictionary is given in Code Fragment 3.8 for the relaxation zone identified by `name0`:

Code Fragment 3.8: Overview of the sub-dictionary in `waveProperties.input`.

```
name0Coeffs
{
    // Insert all required data for any given wave type

    // Insert all required data for any given relaxation zone definition
}
```

The details on wave types can be found in Section 4. The details on relaxation zones and their definition can be found in Section 2.7.

3.1.2.3 Example

An example of a computational domain with two relaxation zones is shown in Code Fragment 3.9. The leftmost relaxation zone is used to generate a second order wave and the rightmost relaxation zone is used to absorp the wave.

The applied solver is `waveFoam`, why the keywork `porosityModel` is commented.

3.1.3 relaxationZoneLayout

The utility `relaxationZoneLayout` can be used to study the layout of the relaxation zone. The utility outputs three fields to the `0`-folder:

- `relaxationZoneLayout`: This field shows the spatial extend of the relaxation zones. The value of -1 is outside of any relaxation zone. The first relaxation zone is given a value of 0, the second a value of 1, etc.
- `relaxationZoneSigmaValue`: This is the spatial distribution of the σ-weight for the relaxation zones.
- `relaxationZoneWeightOnComputed`: The shows the resulting weight that is applied to the computed field, i.e. a value of 1 means that the computed field is *not* altered.

Code Fragment 3.9: Example of `waveProperties.input`.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      waveProperties.input;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//porosityModel   jensenJacobsenChristensen2014;

seaLevel        0.00;

relaxationNames (inlet outlet);

initializationName outlet;

inletCoeffs
{
    waveType    stokesSecond;
    height      0.1;
    period      3.0;
    depth       1.0;
    phi         0.0;
    direction   (1.0 0.0 0.0);
    debug       off;
    Tsoft       2.0;

    relaxationZone
    {
        relaxationScheme Spatial;
        relaxationShape  Rectangular;

        relaxType        INLET;
        startX           (-15.0 0.0 -1.0);
        endX             (-10.0 0.0  1.0);
        orientation      (  1.0 0.0  0.0);
    }
};

outletCoeffs
{
    waveType    potentialCurrent;
    U           (0 0 0);
    Tsoft       2;

    relaxationZone
    {
        relaxationScheme Spatial;
        relaxationShape  Rectangular;

        relaxType        OUTLET;
        startX           ( 10.0 0.0 -1.0);
        endX             ( 15.0 0.0  1.0);
        orientation      (  1.0 0.0  0.0);
    }
};
```

3.1.4 setWaveField

The utility `setWaveField` is used to set the initial conditions according to a user defined wave theory. The latter is defined by the keyword `initializationName` in the file `waveProperties.input`, see Section 3.1.2.1. This can be any wave theory, but if the ramping time differs from 0, then the prescribed initial condition will be a horizontal free surface at still water level and the water will be stagnant. If this is the case, the warning in Code Fragment 3.10 is written to standard output:

Code Fragment 3.10: Warning when `Tsoft` differs from 0

```
--> FOAM Warning :
    From function setWaveField::setWaveField(const fvMesh& mesh, \
        volVectorField& U, volScalarField& alpha, volScalarField& p)
    in file preProcessing/setWaveField/setWaveField.C at line 77

    The specified value of Tsoft is non-zero in the waveType: \
         'stokesFirst'
    specified in the sub-dictionary waveProperties::inletCoeffs

    Consequently, the initialised 'wave field' is set to a horizontal \
        free surface with zero velocity.
```

3.1.5 faceSetToSTL

The modelling of permeable structures, such as breakwaters or open filters, requires that the user defines the location of the permeable structure is located. There are several ways of identifying the cell indices, one of these is the usage of (non-convex) STL-surfaces and the `cellSet` (in foam-extend) or `topoSet` (in OpenFoam) utilities. An STL-surface is essentially a triangulation of a closed surface. The utility `faceSetToSTL` is a tool that creates such a STL-surface. The tool is only easily applicable for simple structures.

The input to the case is the file `stlDefinitions` which is placed in `<casePath>/constant/triSurface`. There is an example located in `<waves2Foam>/applications/utilities/preProcessing/faceSetToSTL`.

There are two ways of defining the STL-surface, namely by defining all the faces of the surface in a manner similar to `blockMesh` or by extruding one face along a given direction (Code Fragment 3.11).

Code Fragment 3.11: Two methods to generate an STL-surface of a unit square.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      stlDefinitions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

unit1
{
    points 4((0 0 0) (1 0 0) (1 1 0) (0 1 0));
    faces 1((0 1 2 3));

    extrude true;
    extrudeVector (0 0 1);
}

unit2
{
    points 8((0 0 0) (1 0 0) (1 1 0) (0 1 0)
             (0 0 1) (1 0 1) (1 1 1) (0 1 1));
    faces 6((0 3 2 1) (4 5 6 7) (2 3 7 6)
            (0 1 5 4) (1 2 6 5) (4 7 3 0));
}
```

The STL-surfaces are written to the folder, where `stlDefinitions` is placed.

3.1.6 sampleIncidentWaveField

The utility `sampleIncidentWaveField` allows for a fast evaluation of the incident wave field based on its *algebraic* form. The code snippet shown in Code Fragment 3.12 is inserted into `waveProperties.input`, followed by execution of the commands shown in Code Fragment 3.13.

The output is an evaluation of the theoretical form of the incident wave field evaluated in the specified points. The surface elevation is evaluated for all *active* relaxation zones. The data is

Code Fragment 3.12: Instructions for `sampleIncidentWaveField` into `waveProperties.input`

```
sampleIncidentWaveField
{
    deltaT    <scalar>; // [s]
    endTime   <scalar>; // [s]

    points N((x1 y1 z1) (x2 y2 z2) ... (xN yN zN));
}
```

outputted in the folder `syntheticWaveField` and the formatting is identical to the formatting from the regular surface elevation tool.

Code Fragment 3.13: Use of `sampleIncidentWaveField`

```
$ setWaveField
$ sampleIncidentWaveField
```

3.2 Run-time sampling

3.2.1 Numerical wave gauges (surfaceElevation)

For run-time sampling of the surface elevation, see Section 3.1.1.3.

3.2.2 Overtopping

The process of overtopping is the amount of water that is overtopping a structure. Overtopping at impermeable and permeable structures is studied with waves2Foam by Jensen et al. (2014).

This functionality should be used in run-time, because the process of overtopping is very rapid, so it is very unlikely that the process is captured at a post-processing step. Consequently, it is recommended to evaluate the overtopping every single time step.

The following sections describe the implementation, the usage and the output.

3.2.2.1 Implementation

During the simulation the following types of face fluxes are available:

- ϕ in [m^3/s] is the flux of fluid across a face;
- ϕ_ρ in [kg/s] is the flux of fluid across a face multiplied the density of the fluid;
- ϕ_F in [m^3/s] is the flux of fluid cross a face multiplied with the indicator function.

While a combination of ϕ and ϕ_F would be perfect to evaluate the flux of water across a face, this is not possible, because ϕ_F is not available throughout the entire time step; consequently, it is not available when the function objects are evaluated. Therefore, the flux of water is estimated with the use of ϕ and ϕ_ρ instead.

In the solution to the advection of the indicator function, the following relationship is used:

$$\phi_\rho = (\rho_{F=1} - \rho_{F=0})\phi_F + \rho_{F=0}\phi \tag{3.1}$$

Consequently, ϕ_F can be estimated as follows

$$\phi_F = \frac{\phi_\rho - \rho_{F=0}\phi}{\rho_{F=1} - \rho_{F=0}} \tag{3.2}$$

Knowing the flux of water (assuming that the fluid is water, when $F = 1$), if is now possible to evaluate the overtopping over a set of faces, $\mathcal{F}$:

$$\mathbf{q} = \sum_{f \in \mathcal{F}} \phi_{F,f} \frac{\mathbf{S}_f}{\|\mathbf{S}_f\|_2} \tag{3.3}$$

where $\mathbf{q}$ is the volume flux in [m^3/s] and $\mathbf{S}_f$ is the non-unit normal vector to the face. Here, ϕ_F is positve in the direction of the normal vector and negative in the opposite direction, so the combination gives the directional overtopping over a set of faces.

Nota Bene 3.1: Regarding estimate of overtopping

It is stressed that ϕ_F is only estimated, because ϕ and ϕ_ρ are not based on the same volume flux at the time of evaluating ϕ_F for the overtopping.

3.2.2.2 Usage

The function-object definition in Code Fragment 3.14 must be inserted in the `controlDict`.

Code Fragment 3.14: Run-time sampling of overtopping

```
overtopping
{
    type overtopping;
    functionObjectLibs ("libwaves2FoamSampling.so");

    outputControl   timeStep;
    outputInterval  1;
}
```

There is no explicit statement on which `faceZones` to operate, because the functionality operates and *all* faceZones called `overtopping*`, i.e. the name merely needs to begin with overtopping.

The face sets are created with the utility `faceSet` for older versions of OpenFoam and (currently) all versions of the foam-extend branch. For newer versions of OpenFoam the utility `topoSet` should be used to create face sets. The face sets are subsequently turned into zones (Code Fragment 3.15).

Code Fragment 3.15: Face zone for overtopping

```
> setsToZones -noFlipMap
```

The reader is referred to the templates for `faceSetDict` and `topoSetDict` that are available with the distribution of the source code on how to use `faceSet` and `topoSet` respectively.

3.2.2.3 Output

The overtopping is outputting in the file `<casePath>//overtopping/<startTimeName>/overtopping.dat`. The use of `<startTimeName>` in the location means that it is possible to restart a simulation without loosing data from the first part of a simulation.

The formatting of `overtopping.dat` is given in Code Fragment 3.16.

Code Fragment 3.16: Formatting of the overtopping data

```
Time: <faceZoneName_0> <faceZoneName_1>   ...   <faceZoneName_N>
t0    <vector>         <vector>           ...   <vector>
t1    <vector>         <vector>           ...   <vector>
...   ...               ...               ...    ...
tM    <vector>         <vector>           ...   <vector>
```

Here, the quantity `t0` is the time instance and `<vector>` is the instantaneous overtopping volume **q**, i.e. it is not an accumulated quantity.

3.3 Post-processing utilities

3.3.1 Numerical wave gauges (surfaceElevation)

Discrepancies have been observed between run-time and post-processing evaluation of the surface elevation with the `surfaceElevation` utility. The run-time results appear to be the correct values, consequently, it is discouraged to use `surfaceElevation` as a post-processing utility.

3.3.2 postProcessWaves2Foam

The utility `postProcessWaves2Foam` is intended for easy post-processing of the run-time sampled properties such as the surface elevation, forces and overtopping.

`postProcessWaves2Foam` consists of three overall elements, namely (i) read the raw data streams, (ii) interpolate the data stream to an equidistant time axis and (iii) process the data. The interpolation is needed for some of the post-processing elements such as the decomposition into incident and reflected wave components for irregular waves, which is essentially based on a FFT.

Warning 3.2: Post-processing

The output format of probes, forces and moments changes significantly between individual versions of OpenFoam. Consequently, it is recommended only to apply `postProcessWaves2Foam` for the processing of output files native to waves2Foam (surface elevastion and overtopping).
The reading of other sources will be removed in future versions of waves2Foam.

The following processing types are currently available:

- Reading raw data
  - `rawSurfaceElevation` reads the surface elevation signal on a - potentially - non-equidistant time axis.
  - `rawAlphaProbes` reads the data from probes of the void-fraction ratio on a - potentially - non-equidistant time axis.
  - `rawVelocityProbes` reads the data from probes of the velocity on a - potentially - non-equidistant time axis.
  - `rawOvertopping` reads the overtopping data on a - potentially - non-equidistant time axis.
  - `rawForcesAndMoments` reads the data on forces and moments on a - potentially - non-equidistant time axis.
- Reading interpolated data
  - `interpolateSurfaceElevation` reads and interpolates the surface elevation signal onto an equidistant time axis.
  - `interpolateAlphaProbes` reads and interpolates probes of the void-fraction ratio onto an equidistant time axis.
  - `interpolateVelocityProbes` reads and interpolates probes of the velocity onto an equidistant time axis.
  - `interpolateOvertopping` reads and interpolates the overtopping data onto an equidistant time axis.
  - `interpolateForcesAndMoments` reads and interpolates the forces and moments onto an equidistant time axis.
- Data handling
  - `write2Ascii` writes the data stream to an easily accessible ASCII-format.
  - `writeIndexLocation` writes the index location of the surface elevation data.
  - `removeData` removes data from intermediate steps
  - `removeSubDirs` removes subdirectories from processing steps.
- Data processing
  - `trapz` performs a trapezoidal integration in time.
  - `cumTrapz` performs a cummulative trapezoidal integration in time.
  - `zeroCrossing` performs a zero-crossing analysis of a given time series.
  - `ensembleAverage` performs an ensemble averaging analysis of a given time series.
  - `powerSpectraLS` performs a power spectrum analysis based on a least-squares technique; in particular useful for regular waves.
  - `reflectionAnalysis2DLS` performs a reflection analysis based on a least-squares techniques; in particular useful for regular waves (Jacobsen et al., 2012).

- `powerSpectraFFT` performs an evaluation of the power spectrum for a given time series with the use of the Fourier transform; in particular useful for irregular waves.
- `reflectionAnalysis2DFFT` performs a decomposition into incident and reflected waves based on a Fourier transport; in particular useful for irregular waves (Zelt and Skjelbreia, 1992).

Besides this list, the utility is only documented through its application in some of the utilities.

CHAPTER 4

Wave Theories

This chapter discusses the wave theories available in waves2Foam. The information for the wave theories are given in the file `waveProperties.input` and the required set of files to run external wave theories from third-party software source (Sections 4.2.8 and 4.3).

The required information in `waveProperties.input` are either given in a sub-dictionary for a given name of a boundary or as global information. The overall organisation of `waveProperties.input` is described in Section 3.1.2.

4.1 Validity of wave theories

The choice of an appropriate wave theory should be based on a few considerations:

- Is the wave theory valid in the range of wave heights, wave periods and water depths to be studied?
- Is an experimental study sought reproduced? If yes, which wave theory did they apply in the experimental study?

With respect to the first question, then there are several sources available, which treat the validity of wave theories. Regarding the second question, then it is important to stay as close as possible to the experimental study to obtain accurate results. Here, either the appropriate wave theory should be chosen or the actual paddle signal should be used as a boundary condition (see e.g. Higuera et al., 2015; Jacobsen et al., In review, for the latter).

4.2 Algebraic wave theories

The algebraic wave theories are covered in this section. This refers to wave theories that are given in algebraic form. Each subsection contains information on the required input data for a particular wave theory and one or more references to the implementation.

The settings are procided in "Code Fragments", where both the mandatory and non-mandatory parameters are provided. The default values for non-mandatory parameters are specified.

4.2.1 Regular wave theories

4.2.1.1 Stokes first order

The first order Stokes wave theory is encountered in a multiple of sources (Dean and Dalrymple, 1991; Svendsen and Jonsson, 1982). The latter source is used the waves2Foam-implementation. The settings are provided in Code Fragment 4.1.

4.2.1.2 Standing Stokes first order

The first order Stokes standing wave is a superposition of two waves of equal amplitude that propagate in opposite directions. The implementation into waves2Foam follows Dean and Dalrymple (1991). The settings are provided in Code Fragment 4.2.

Code Fragment 4.1: Settings Stokes First Order

```
waveType   stokesFirst;
depth      <scalar>;  // [m]
period     <scalar>;  // [s]
direction  <vector>;  // [-]
phi        <scalar>;  // [radians]
height     <height>;  // [m]
Tsoft      <scalar>;  // [s] Default value: period
```

Code Fragment 4.2: Standing Stokes First Order

```
waveType   stokesFirstStanding;
depth      <scalar>;  // [m]
period     <scalar>;  // [s]
direction  <vector>;  // [-]
phi        <scalar>;  // [radians]
height     <scalar>;  // [m]
Tsoft      <scalar>;  // [s] Default value: period
```

4.2.1.3 Stokes second order

The second order Stokes theory is a perturbation extension to the first order theory. The extension allows for a finite wave height and thus extends the range of validity of the wave in terms of nonlinearity. The implementation follows the description in Svendsen and Jonsson (1982). The settings are provided in Code Fragment 4.3.

Code Fragment 4.3: Stokes Second Order

```
waveType   stokesSecond;
depth      <scalar>;  // [m]
period     <scalar>;  // [s]
direction  <vector>;  // [-]
height     <scalar>;  // [m]
phi        <scalar>;  // [radians]
Tsoft      <scalar>;  // [s] Default value: period
debug      <bool>;    // [-]
```

4.2.1.4 Modulated second-order Stokes wave

The modulated second order Stokes wave "theory" is an ad.hoc. adjustment to second order wave theory, with a wave height that is modulated in time. It should be stressed that this wave theory is not an accurate second order wave theory. It has been applied to study the effect of a small irregularity in a incident wave on cross shore morphodynamics (Jacobsen and Fredsøe, 2014a,b). The modulation to the wave height is as follows:

$$H_m = H_0 \left(1 + \epsilon \sin \frac{1}{N}(\omega t - \mathbf{k} \cdot \mathbf{x})\right) \tag{4.1}$$

Here, H_m is the modulated wave height, H_0 is the mean wave height, ϵ is the magnitude of the modulation, N is an integer, ω is the cyclic frequency of the primary wave and $\mathbf{k}$ is the wave number vector. The settings are provided in Code Fragment 4.4.

4.2.1.5 Stokes fifth order

The fifth order Stokes theory further extend the validity for larger nonlinearities of the waves in comparison with e.g. the second order Stokes theory, see (Fenton, 1985, 1990). The largest difference is that the net flux of water in terms of the Stokes drift is needed in order to evaluate the wave properties. Please note that the perturbution theory is not always applicable (Section 4.1. The settings are provided in Code Fragment 4.5.

Code Fragment 4.4: Modulated Second Order Stokes

```
waveType   stokesSecondModulation;
depth      <scalar>;  // [m]
period     <scalar>;  // [s]
direction  <vector>;  // [-]
height     <scalar>;  // [m]
phi        <scalar>;  // [radians]
epsilon    <scalar>;  // [-]
modN       <scalar>;  // [-] Number of waves per modulation
Tsoft      <scalar>;  // [s] Default value: period
debug      <bool>;    // [-]
```

Code Fragment 4.5: Stokes Fifth Order

```
waveType    stokesFifth;
height      <scalar>;  // [m]
period      <scalar>;  // [s]
depth       <scalar>;  // [m]
stokesDrift <scalar>;  // [m/s]
direction   <vector>;  // [-]
Tsoft       <scalar>;  // [s] Default value: period
phi         <scalar>;  // [radians]
```

4.2.1.6 First-order cnoidal theory

Cnoidal wave theory is intended for shallower waters than class of Stokes theories and the underlying assumption in the derivation is that the ratio h/L (h is the water depth and L is the wave length) is a small number. The waves2Foam-implementation follows the description in Svendsen and Jonsson (1982). The settings are provided in Code Fragment 4.6.

Code Fragment 4.6: First Order Cnoidal Theory

```
waveType    cnoidalFirst;
height      <scalar>;     // [m]
depth       <scalar>;     // [m]
period      <scalar>;     // [s]
direction   <vector>;     // [-]
Tsoft       <scalar>;     // [s] Default value: period
```

The key parameter for cnoidal wave theory is the shape parameter m, which needs to be strictly less than 1. Consequently, the utility `setWaveParameters` automatically increases the write precision of m to 14 digits to ensure that m is written to `waveProperties` as a value strictly smaller than 1.

4.2.1.7 Stream function wave

Stream function wave theory is a nonlinear solution to the steady wave problem in arbitrary depth. The theory is - amongst others - described in Rienecker and Fenton (1981); Fenton (1988). This wave theory is generally applicable as long as the wave height is below the breaking criterion (Section 4.1). The settings are provided in Code Fragment 4.7.

The utility used to evaluate the coefficients is based on the original source code provided in Fenton (1988)[1].

4.2.2 Bichromatic wave theories

4.2.2.1 First-order bichromatic wave

A bi-chromatic wave is a wave with two frequency components. The first order wave is a simple linear super-position of two Stokes first order waves with two different wave heights (Madsen and Fuhrman, 2006). The settings are provided in Code Fragment 4.8.

[1] The present version was originally typed by Jesper Skourup (DHI) and it was later ammended by Harry Bingham (Technical University of Denmark). The source code was made available to waves2Foam by Henrik Bredmose (Technical University of Denmark).

Code Fragment 4.7: Stream Function Theory. Note the `EITHER-OR` structures.

```
waveType       streamFunction;
height         <scalar>;  // [m]
depth          <scalar>;  // [m]
N              <scalar>;  // [-] Number of components
Niter          <scalar>;  // [-] Number of iterations
phi            <scalar>;  // [radians]
direction      <vector>;  // [-]

EITHER:
specifyPeriod  false;     // [boolean]
waveLength     <scalar>;  // [m]
OR:
specifyPeriod  true;      // [boolean]
period         <scalar>;  // [s]

EITHER:
specifyEuler   true;      // [boolean]
eulerVelocity  <scalar>;  // [m/s]
OR:
specifyEuler   false;     // [boolean]
stokesVelocity <scalar>;  // [m/s]
```

Code Fragment 4.8: First Order Bichromatic

```
waveType    bichromaticFirst;
depth       <scalar>;  // [m]
period1     <scalar>;  // [s]
period2     <scalar>;  // [s]
direction1  <vector>;  // [-]
direction2  <vector>;  // [-]
height1     <scalar>;  // [m]
height2     <scalar>;  // [m]
phi1        <scalar>;  // [radians]
phi2        <scalar>;  // [radians]
Tsoft       <scalar>;  // [s] Default value: max(period1, period2)
```

4.2.2.2 Second-order bichromatic wave

The second order bichromatic wave is similar to the first order variant, but it includes all the wave-wave interactions between two waves with different periods and propagation directions, (Madsen and Fuhrman, 2006). The settings are provided in Code Fragment 4.9.

Code Fragment 4.9: Second Order Bichromatic

```
waveType    bichromaticSecond;
depth       <scalar>;  // [m]
period1     <scalar>;  // [s]
period2     <scalar>;  // [s]
direction1  <vector>;  // [-]
direction2  <vector>;  // [-]
height1     <scalar>;  // [m]
height2     <scalar>;  // [m]
phi1        <scalar>;  // [radians]
phi2        <scalar>;  // [radians]
Tsoft       <scalar>;  // [s] Default value: max(period1, period2)
```

4.2.3 First-order irregular waves

The first order irregular wave theory is a simple linear superposition of first order Stokes waves, where the first order Stokes theory is extrapolated to the free surface:

$$\eta = \sum_{i}^{N} a_i \cos(\omega_i t - \mathbf{k}_i \cdot \mathbf{x} + \varphi_i) \tag{4.2}$$

Here, η is the surface elevation, N is the number of wave components, a_i is the amplitude of the i'th wave component, ω is the cyclic frequency, $\mathbf{k}$ is the wave number vector and φ is a phase (random or focused).

In order to give the information for the irregular waves, there are currently three classes of information needed:

- Spectral shape
- Spectral discretisation
- Phases

The shape determines the variation of a_i with ω_i, the discretisation determines the discrete representation of ω_i and the phases determines the values of φ_i.

It is possible to extend the present options by modifications of the run-time selectable classes located in `<waves2Foam_src>/src/waves2FoamProcessing/preProcessing/setWaveProperties/irregular/waveSpectra`. The three classes of information are implemented in the sub-directories `spectra`, `frequencyAxis` and `phases`.

The following paragraphs describe the various settings needed to prescribe an incident irregular wave field. Some common settings are provided in Code Fragment 4.10.

Code Fragment 4.10: Irregular waves: Common settings

```
waveType      irregular;
N             <label>;   // [-] Number of wave components
Tsoft         <scalar>;  // [s] Ramping time
```

4.2.3.1 Spectral shape

The following spectral shapes are currently implemented:

- JONSWAP
- Pierson-Moskowitz

Information on these spectra can be found in many standard introductions to irregular wave theory, e.g. Holthuijsen (2007). The settings for JONSWAP are provided in Code Fragment 4.11 and the settings for Pierson-Moskowitz are provided in Code Fragment 4.12.

Code Fragment 4.11: Irregular waves: JONSWAP settings

```
spectrum      JONSWAP;
Hs            <scalar>; // [m] Spectral wave height
Tp            <scalar>; // [s] Peak wave period
gamma         <scalar>; // [-] Peak enhancement factor
depth         <scalar>; // [m]
direction     <vector>; // [-]
```

Code Fragment 4.12: Irregular waves: Pierson-Moskowitz settings

```
spectrum      PiersonMoskowitz;
Hs            <scalar>; // [m] Spectral wave height
Tp            <scalar>; // [s] Peak wave period
depth         <scalar>; // [m]
direction     <vector>; // [-]
```

4.2.3.2 Spectral discretisation

There are currently two discretisation methods available for the frequequency axis:

- `equidistantFrequencyAxis`
- `cosineStretchedFrequencyAxis`

The latter produces a stretching of the frequency axis towards the peak of the spectrum. This stretching (and any other non-equidistant stretching) greatly improves on the statistical properties of resulting time series for the surface elevation, see Appendix B for details.

The frequency discretisation is described in a sub-dictionary inside of the definition of the irregular wave (Code Fragment 4.13). The keyword `frequencyAxis` refers to the discretisation options metioned in the list above.

Code Fragment 4.13: Irregular waves: Frequency discretisation

```
frequencyAxis
{
    discretisation       <frequencyAxis>;

    lowerFrequencyCutoff <scalar>;  // [Hz]. Default: 1/(3Tp)
    upperFrequencyCutoff <scalar>;  // [Hz]. Default: 3/Tp

    writeSpectrum        <bool>;    // Write the target spectrum (f, S)
                                    // and not just (f, amp)
}
```

4.2.3.3 Phases

There are two types of phasing methods available and these are is either a random phase or a phase focusing for a certain location, $\mathbf{x}_0$ at a certain time instance t_0. The settings for the random phases are provided in Code Fragment 4.14 and the settings for the phase focusing are provided in Code Fragment 4.15.

Code Fragment 4.14: Irregular waves: Random phases

```
phaseMethod        randomPhase; // Default value
seedForRandomPhase <label>;     // No required
```

If a value for the seeding is not provided (Code Fragment 4.14), then the seeding for the random number generator is based on the clock time. Applying a fixed seeding ensures that the results can be reproduced.

Code Fragment 4.15: Irregular waves: Focused phases

```
phaseMethod focusingPhase;
focusTime   10.0;           // [s]
focusPoint  (0 0 0);        // [m]
```

Nota Bene 4.1: A note on phase focusing

The phases are evaluated to obtain phase focusing based on linear wave theory. Due to amplitude dispersion, a bit of tweeking of the focus point and focus time must be expected for large wave heights.

4.2.3.4 Example

A compilation of all of the various components is exemplified in Code Fragmentcode:irregularExample. This example produces an irregular wave with 100 wave components described with the JONSWAP spectrum with a peak enhancement factor of 3.3. The spectral wave height is 1 m, the

peak wave period is 10 s and the water depth if 10 m. The phases are random (but reproducible) with a fixed seeding of 10. The frequency axis is equidistantly discretised and user defined frequency cut-offs are specified.

Code Fragment 4.16: Irregular waves: Example

```
// Define the wave discretisation and ramping
waveType           irregular;
N                  100;
Tsoft              10;

// Define the phases
phaseMethod        randomPhase;
seedForRandomPhase 10;

// Define the spectrum
spectrum           JONSWAP;
Hs                 1;
Tp                 10;
gamma              3.3;
depth              10;
direction          (1 0 0);

frequencyAxis
{
    discretisation       equidistantFrequencyAxis;

    lowerFrequencyCutoff 0.01;
    upperFrequencyCutoff 0.4;

    writeSpectrum        false;
}
```

4.2.4 Second-order irregular waves

Second order irregular waves are currently not implemented, because the computational efforts are likely to be prohibitively expensive (Appendix B). Second order irregular wave theory includes wave-wave interaction terms and the theory are described in e.g. Sharma and Dean (1981); Madsen and Fuhrman (2012).

In the work by Madsen and Fuhrman (2012) it was shown that some of the coefficients in the perturbation solution to the third order irregular wave theory become unbounded and violates the perturbation assumption. This means that there does not exist a (bounded) perturbation solution for the irregular wave problem beyond second order. Therefore numerical models are required to specify nonlinear, irregular incident waves (Section 4.3.3).

4.2.5 Potential current

The "wave theory" `potentialCurrent` is a means of introducing a current that is uniform over the depth, hence the name. This is typically used for outlet relaxation zones, where the velocity vector is set of $\mathbf{0}$. The settings are provided in Code Fragment 4.17.

Code Fragment 4.17: Potential current

```
waveType   potentialCurrent;
U          <vector>;  // [m/s]
Tsoft      <scalar>;  // [s]
```

4.2.6 Solitary wave theories

4.2.6.1 First-order solitary wave

A solitary wave is a single wave crest and the current implementation follows that presented in Svendsen and Jonsson (1982). The settings are provided in Code Fragment 4.18.

Here, the keyword `x0` is the location of the crest at $t = 0$ s.

Code Fragment 4.18: Solitary First Order

```
waveType   solitaryFirst;
height     <scalar>;  // [m]
depth      <scalar>;  // [m]
direction  <vector>;  // [-]
x0         <vector>;  // [m]
```

4.2.6.2 Chappelear (1962)

The higher-order solitary wave theory according to Chappelear (1962) is also implemented. The settings are provided in Code Fragment 4.19.

Code Fragment 4.19: Solitary wave (Chappelear, 1962)

```
waveType   chappelear1962;
height     <scalar>;  // [m]
depth      <scalar>;  // [m]
direction  <vector>;  // [-]
x0         <vector>;  // [m]
```

Here, the keyword `x0` is the location of the crest at $t = 0$ s.

4.2.7 Combined waves

The use of the wave theory `combinedWaves` allows for the combination of any of the wave theories described in this chapter. Merely specify a word list of other sub-dictionaries in the file `waveProperties.input`. The settings are provided in Code Fragment 4.20.

Code Fragment 4.20: Linear Superposition

```
waveType          combinedWaves;
combinedWaveNames <wordList>;
```

Please note that a combination of e.g. `potentialCurrent` with `stokesFirst` does not affect the evaluation of the wave number, i.e. the Doppler Effect is not included. Tailored wave theories are required to correctly model wave-current interaction.

The combined waves is convenient, if the same properties are required on multiple boundaries.

4.2.8 External source

It is possible to couple waves2Foam to an external source that provides the information on the velocity field and the surface elevation.

This wave theory has a default value, which means that no external wave source is defined, i.e. an external wave theory is always constructed, but it is not used. It is only possible to have *one* external source in a simulation. The settings require both a globel setting and settings required for the individual boundaries (Code Fragment 4.21). Besides these settings, there are settings specific to the external wave model and these are specified in Section 4.3.

Code Fragment 4.21: Use of `externalSource`

```
// Global setting
externalForcing <string>; // Default: emptyExternal;

// Settings for individual boundary
<name>Coeffs
{
    waveType        externalSource;
}
```

Nota Bene 4.2: Always specify an external source (if used)

If `externalSource` is given as a waveType, a valid external source has to be provided, since the default type `emptyExternal` is not accepted.

4.3 External wave theories

One of the wave theories is called `externalSource` (see Section 4.2.8). This theory allows for a smooth coupling between waves2Foam and any algebraic or non-algebraic method for computing the wave field; this could for instance be the coupling between an external numerical solution of the nonlinear wave problem and waves2Foam. The available methods are described in the following, however, the limitations is largely due to the limitations in interfacing with other models.

All external wave theories have one thing in common: The control of the coupling is specified through (i) files native for the external wave theory and (ii) the sub-dictionary called `externalForcingCoeffs`.

4.3.1 Empty external method

The method is a dummy method that cannot be used. It is needed to construct an external source method, when the algebraic methods are used, which explains the existence of this theory. It is not necessary to specify the settings (Code Fragment 4.22) in `waveProperties.input`, since it is the default value for `externalForcing`.

Code Fragment 4.22: Empty external source (Default)

```
// Global setting
externalForcing emptyExternal;
```

4.3.2 Fast summation of irregular waves

This external wave theory produces identical results to the ordinary irregular wave implementation described in Section 4.2.3; the definitions of frequency discretisation, phasing methods and spectral shape are identical.

The main difference between the methods is that `irregularFast` performs a lot of preprocessing to lower the amount of time spend to evaluate trigonometric and hyperbolic functions. These pre-computed sets of data are stored in arrays for unique values of x, y and z. If needed, itnerpolation will be performed between these values. The consequence is a considerable speed-up (Appendix B).

The global settings are provided in Code Fragment 4.23. Furthermore, the sub-dictionary information must be specified in a sub-dictionary called `externalForcingCoeffs`, where the content is the definition of the irregular wave as exemplified in Code Fragment 4.16.

Code Fragment 4.23: Fast summation of irregular wave theories

```
// Global settings
externalForcing   irregularFast;
searchTolerance   <scalar>;       // [m]
ignoreMeshMotion  <boolean>;      // [-]
```

The method is not intended to be used with mesh motion, but in certain cases the mesh motion is the relaxation zone is so small that it is safe to ignore that effect. If `ignoreMeshMotion` is set to `false/no`, then the simulation will stop in case of mesh motion.

The keyword `searchTolerance` is related to the interpolation routine between the unique values of x, y and z. If a unique point within the distance of `searchTolerance` is found, then no interpolation is executed.

4.3.3 OceanWave3D

waves2Foam is coupled with the third-party software OceanWave3D, which is able to evaluate fully nonlinear wave propagation (Engsig-Karup et al., 2009). The coupling between Ocean-

Wave3D and waves2Foam is described in Paulsen et al. (2014a) and the current implementation is streamlined with the general external source functionality in waves2Foam by Bo T. Paulsen and Niels G. Jacobsen. The settings are provided in Code Fragment 4.24.

Nota Bene 4.3: Installation of OceanWave3D

OceanWave3D is installed alongside waves2Foam, why the user does not have to do any additional steps. The downloaded source code also contains a GUI, which allows for easy setup of 2DV simulations with OceanWave3D. The GUI is not compiled as part of waves2Foam.

Code Fragment 4.24: Coupling with OceanWave3D

```
// Global setting
externalForcing oceanWave3D;

externalForcingCoeffs
{
    waveType oceanWave3D;

    // Define the intervals for the OpenFoam calculations
    nIntervals 1;
    startTimes nonuniform List<scalar> 1(5);
    endTimes nonuniform List<scalar> 1(20);

    // Should the interval be ramped?
    rampInterval off;

    // Name of the sub-dictionary (without Coeffs), where the external-
    // Source definition is given. Is needed, when the mapping OCW3D to
    // OpenFoam is carried out.
    mappingZone inlet;
};
```

4.3.3.1 Output

The output from OceanWave3D consists of three elements:

- `Kinematics*` and `*.fort` files. The output in these files cannot be used, because the time stepping is controlled by OpenFoam, hence the time stamps on these files are wrong.
- `waveGauges.dat` contains ASCII-formatted wave gauge data, which comes in the same format as that for the standard surfaceElevation data in waves2Foam. The time stamp is correct.
- Hot-start files are written in synchronisation with the OpenFoam results. The files are outputted to the folder `<casePath>/OCW3Dhotstart`. The naming convection is `OceanWave3D.-<timeIndex>`, where `<timeIndex>` is the time index in the OpenFoam simulation. This index can be retrieved in the file `<timeName>/uniform/time`.

4.4 Extensions with new theories

Extension of waves2Foam with new wave theories is relatively straightforward for both the algebraic and the external wave theories. The following two sections describes the approach to develop new wave theories for each of these two classes.

4.4.1 Extension: Algebraic wave theory

In the directory `<waves2Foam_src>/doc/templateWaveTheory` there is a small script called `makeNewWaveTheory`. Execute this script (Code Fragment 4.25) to create a new wave theory called `myWaveFormulation`:

The script creates output as shown in Code Fragment 4.26.

It is merely a matter of following this step-by-step guideline to implement a new wave theory.

Code Fragment 4.25: Create a new algebraic wave theory

```
> ./makeNewWaveTheory myWaveFormulation
```

Code Fragment 4.26: Create a new wave theory - Output

```
A directory called 'myWaveFormulation' with [C,H]-files is created.

Steps to finalise:
1. Copy the directory to the correct wave type in waves2Foam/src/ \
   waveTheories
2. Insert the needed variables and expressions for eta, U, etc.
3. Add a line in waves2Foam/src/Make/files to include the new theory in
   the compilation.
4. Done.
```

4.4.2 Extension: External source

There is no easy development guide for the coupling with an external source. This is because the class will depend on the interface to the external source. Some relevant thoughts are:

- Is it possible to start the external source at a time different from $t = 0$ s? If not, it should be possible to take several time steps in the external source before activating OpenFoam simulation, if the latter needs restarting;
- If it is possible to restart the external source, then make sure to save data for the external source at the same time instances as data is saved in the OpenFoam simulation;
- For a proper integration of the external source with waves2Foam, it should be possible to control the time stepping of the external source through OpenFoam;
- For an optimal integration of the external source with waves2Foam, the external source should be compatible with a variable time steps;
- Alternatively, a time integration scheme between an equidistant time stepping in the external source and OpenFoam should be implemented and consequently a time control of the external source that allows for the external force to be ahead in time by one or more time steps.

CHAPTER 5

Solvers

5.1 waveFoam

The waveFoam solver is a dedicated solver to the study of wave interaction and propagation in a static computational domain. There may be one or more fixed structures, but permeable layers are not included.

5.2 porousWaveFoam

The porousWaveFoam solver is similar to waveFoam (Section 5.1) with the addition that one can specify one or more permeable layers.

5.3 waveDyMFoam (moving meshes)

waves2Foam is not distributed with a solver for moving meshes and there is similarly not any tutorials that shows the use of waves2Foam with moving meshes. There is, however, no limitations in the waves2Foam framework that prevents a merging of the `interDyMFoam` and waves2Foam.

While there is no solver available, waves2Foam does take account of the following:

- Moving meshes: The weights in the relaxation zone are re-calculated every time step in case of moving meshes.
- Topological changes: If there are topological changes in the computational mesh due to e.g. mesh refinement or addition or removal of computational cells, the location of the relaxation zones are recomputed every time step.

waves2Foam has already been used with moving meshes in several works (see Section **??**).

> **Warning 5.1:** Online documentation to create `waveDyMFoam`.
>
> Do not use the guide on the Wiki-page or on `https://sites.google.com/site/jordimuela/openfoam/how-to-couple-waves2foam-with-dynamic-mesh-motion`, since the guides are outdated.

5.3.1 Modifications to `interDyMFoam`

The following steps can be taken in order to create a solver `waveDyMFoam`:

- Make changes to the interDyMFoam files, see Section 5.3.1.1 for details;
- Update the file `Make/options`, see Section 5.3.1.2 for details;
- Update the file `Make/files`, see Section 5.3.1.3 for details;

5.3.1.1 Changes to files

The Code Fragment 5.1 may be used to display the differences between the `interFoam` and `waveFoam` solvers. This information can be used to make the required modifications to create `waveDyMFoam`:

Code Fragment 5.1: Example code to compare files in order to understand modifications made from `interFoam` to `waveFoam`

```
#!/bin/bash

list=‘ls *C *H‘

for i in $list
do
    echo "============ BEGIN: "$i" ============"

    if [ "$i" == "waveFoam.C" ]
    then
        diff $i $FOAM_APP/solvers/multiphase/interFoam/interFoam.C
    else
        diff $i $FOAM_APP/solvers/multiphase/interFoam/$i
    fi

    echo
    echo "============= END: "$i" ============="
    echo
done
```

Place the script in the `waveFoam` directory for the installed (and sourced) version of waves2Foam. Execute the commands given in Code Fragment 5.2.

Code Fragment 5.2: How to execute the comparison

```
> chmod a+x compareFiles.sh
> ./compareFiles.sh
```

Warning 5.2: Warning concerning the definition of the gravity vector.

It is necessary to create the gravity vector *before* creating the fields in the `createFields.H` file. In some versions of OpenFoam, the gravity vector is create in `createFields.H` after the constructions of the volume fields. The code will compile, but it will not be possible to execute it.
This explains why the gravity vector is moved outside of `createFields.H` and into `waveFoam.C`.

After the required changes have been identified, copy the files belonging to `interDyMFoam` to a folder next to `waveFoam`, incorporate the changes and rename `interDyMFoam.C` to `waveDyMFoam.C`.

5.3.1.2 Details on `Make/options`

Additional lines are needed in the `EXE_INC`-section of `Make/options` (Code Fragment ??) and additional lines in the `EXE_LIBS`-section of `Make/options` (Code Fragment ??). Take care to add the necessary backslash:

Code Fragment 5.3: Additional lines in `Make/options` for `EXE_INC`.

```
-DOFVERSION=<Numerized version number> \
-DEXTBRANCH=<OF-release = 0. FE-release = 1> \
-I$(WAVES_SRC)/waves2Foam/lnInclude \
-I$(WAVES_SRC)/waves2FoamSampling/lnInclude \
-I$(WAVES_GSL_INCLUDE)
```

Code Fragment 5.4: Additional lines in `Make/options` for `EXE_LIB`.

```
-L$(WAVES_LIBBIN) \
-lwaves2Foam \
-lwaves2FoamSampling \
-L$(WAVES_GSL_LIB) \
-lgsl \
-lgslcblas
```

5.3.1.3 Details on `Make/files`

The file `Make/files` should look as in Code Fragment 5.5.

Code Fragment 5.5: Modifications to `Make/files`

```
waveDyMFoam.C

EXE = $(WAVES_APPBIN)/waveDyMFoam
```

CHAPTER 6

Tutorials

6.1 Execute a tutorial

The handling of cross-version compatibility means that several actions are required prior to execution of the tutorial. Due to this, all cases come with an `Allrun`-script. It is simply a matter of executing this script according to Code Fragment 6.1 to run the tutorial. Opening the script furthermore outlines the used commands.

Code Fragment 6.1: Executing a tutorial

```
$ ./Allrun
```

6.2 waveFoam

6.2.1 `standingWave`

This tutorial shows how to handle wave generation and a fully reflecting sea wall utilising the relaxation zone technique.

The following utilities and solver relevant for waves2Foam are used:

- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.2.2 `waveFlume`

This tutorial shows the user how to use the `waveProperties.input` dictionary to set-up the wave properties, relaxation zones and initialization specifications. Furthermore, the tutorial illustrates the physical wave transformation that occurs if an invalid wave theory is used at the inlet (see e.g. experimental work by Chapalain et al., 1992).

The following utilities and solver relevant for waves2Foam are used:

- `waveGaugesNprobes`
- `setWaveParameters`
- `setWaveField`
- `waveFoam`
- `postProcessWaves2Foam`

6.2.3 bejiBattjes

This tutorial provides the experimental setup from Beji and Battjes (1993) and a matlab-script, which can be used to compare the experimental data and the simulation. This tutorial is not optimised to work as validation case.

The following utilities and solver relevant for waves2Foam are used:

- `waveGaugesNprobes`
- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.2.4 couplingOceanWave3D

This tutorial shows how waves2Foam can be coupled with the nonlinear potential wave model for wave transformation: OceanWave3D.

The following utilities and solver relevant for waves2Foam are used:

- `waveGaugesNprobes`
- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.2.5 periodicSolitary

This tutorial initialises the numerical wave tank with solitary wave. The computational domain is cyclic in the direction of wave propagation and there are no relaxation zones. Note that the simulation generates a spurious wave propagating in the *opposite* direction of the solitary wave. This is thought to be due to the low order of the solitary wave theory.

The following utilities and solver relevant for waves2Foam are used:

- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.2.6 3Dwaves

This tutorial shows the use of relaxation zones in 3D. There are 3 inlet relaxation zones and one outlet relaxation zone. In the middle of the wave tank there is a vertical internal wall on which the waves are reflected. The waves, which are generated in the 3 inlet relaxation zones, have different directions relative to the x-axis.

The following utilities and solver relevant for waves2Foam are used:

- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.2.7 squarePile

This tutorial shows how to use the cylindrical relaxation zone. A unidirectional current is the target function all the way around, hence one relaxation zone acts as both generation and absorption. The relaxation zone removes the internally generated surface disturbances. It could be interesting to have it tested for ships waves. Alternatively, a better shape for that purpose, e.g. an elliptical relaxation zone shape could be implemented.

The following utilities and solver relevant for waves2Foam are used:

- `setWaveParameters`
- `setWaveField`
- `waveFoam`

6.3 porousWaveFoam

6.3.1 `porousDamBreak`

This tutorial gives an example of the use of `porousWaveFoam`. The case is of a dambreak through a permeable structure and there are post-processing scripts available to compare with experimental data by Liu et al. (1999). The comparison is reported in Jensen et al. (2014).

The following utilities and solver relevant for waves2Foam are used:

- `setWaveParameters`
- `waveGaugesNProbes`
- `porousWaveFoam`
- `postProcessWaves2Foam`

6.4 Miscellaneous

6.4.1 `relaxationZoneLayout`

This tutorial will show how to use the utility to obtain a visual representation of the relaxation zone(s). This could be helpful to detect potential problems in the setup.

The following utility relevant for waves2Foam are used:

- `relaxationZoneLayout`

CHAPTER 7

Validation

This chapter is intended as a community effort, where all users of waves2Foam can provide small text pieces. The text pieces should put the work with waves2Foam into perspective of the existing literature and present cases, where OpenFoam and waves2Foam are validated. Each section shows the list of author(s) and their affiliation.

How to contribute? Please send text (LaTeX-format), references (bib-format) and graphics[1] (eps-format) through my ResearchGate account:
Niels Gjøl Jacobsen (https://www.researchgate.net/profile/Niels_Jacobsen3).

7.1 Loads on bridge decks

Contributor: Betsy Seiffert (Florida Atlantic University). June 2017

A group of researchers at the University of Hawaii did an extensive study on wave loads on coastal bridges during a study on the vulnerability of coastal bridges to hurricane and tsunami. In Seiffert et al. (2014), Hayatdavoodi et al. (2014), Hayatdavoodi et al. (2015b), and Seiffert et al. (2015), calculations for surface elevation, as well as vertical and horizontal force due to solitary and cnoidal wave impact on a flat plate and bridge model with girders was compared with experimental measurements, for a range of fully elevated to fully submerged bridge elevations. Numerical calculations were performed using OpenFOAM, and Euler's equations were solved in the fluid domain by setting viscosity to approximately zero. Solitary and cnoidal waves were generated using the wave generation and absorption toolbox waves2Foam developed by Jacobsen et al. (2012) and using solutions of the Level 1 Green-Naghdi equations for surface elevation, given in Ertekin and Wehausen (1986), Ertekin and Becker (1998), and Sun (1991), for example. Additionally, effects of air entrapment and venting on forces on a bridge deck was studied in Hayatdavoodi et al. (2014) by first comparing experiments and calculations where air is fully trapped between the girders, and then by adding air relief holes to the bridge deck in numerical calculations such that air can fully escape between the girders and comparing with experiments. For each of these studies, OpenFOAM was able to accurately calculate reflection, transmission and dispersion of waves, as well as vertical and horizontal forces on the flat plate and bridge model.

Using validations from this study, Hayatdavoodi et al. (2015a) assessed the vulnerability of four selected bridges on the island of Oahu, Hawaii, to estimated maximum hurricane waves loads based on simulated hurricane landfall at each bridge location. Calculations of vertical and horizontal force on each selected bridge geometry are made by solving for Euler's equations in OpenFOAM, and by another code which solves for force using the Green-Naghdi equations, and which simplifies the bridge as a fully submerged flat plate. Calculations are compared with theoretical calculations made using linear long-wave theory (Patarapanich, 1984) and some simplified design-type force equations (Douglass et al., 2006; McPherson, 2008; Kulicki and Mertz, 2008, (AASHTO)).

Effects of the compressibility of air trapped between bridge girders, and scaling was studied by Seiffert et al. (2015) by solving for both the compressible and incompressible Euler's equations in OpenFOAM and comparing with experimental measurements. For this study, Seiffert et al. (2015) successfully integrated the waves2Foam toolbox into the compressible version of `interFoam`, namely `compressibleInterFoam`.

Bricker and Nakayama (2014) performed numerical calculations to study the cause of failure of the Utatsu bridge during the 2011 Great East Japan Tsunami. Time series for flow depth and velocity at the bridge location were taken from a large-scale Delft shallow water simulation and applied to a 2D bridge model in OpenFOAM. Using these calculations, the authors determined the cause of failure was

[1] Please make sure that graphics (figures, photos, etc) are *not* under any copyright agreement. If graphics are provided, it is an implicit statement from the authors that the graphics may be included in this manual without any violation of copyrights.

likely due to a combination of factors, include the presence of a nearby sea wall, the seaward inclination of the bridge, entrained sediment and entrapped air between the girders.

7.2 Interaction with coastal structures

Contributor: Niels Gjøl Jacobsen (Deltares). May 2017

The interaction of waves and coastal permeable structures has been studied with several VOF-based numerical models. Examples are those of Skylla (Van Gent et al., 1994), COBRAS (Liu et al., 1999), IH2VOF (Losada et al., 2008; Guanche et al., 2009), ComFlow (Wellens et al., 2010) and lately OpenFoam, where either of the two wave generation toolboxes IHFoam (Higuera et al., 2014) and waves2Foam (Jensen et al., 2014; Jacobsen et al., 2015; Van Gent et al., 2015; Jacobsen et al., In review) were applied.

The modelling of permeable coastal structures has been subject to some debate, and differences in the formulation of the porous medium are observed between the formulations of for instance Wellens et al. (2010); del Jesus et al. (2012); Higuera et al. (2014); Jensen et al. (2014). The differences are amongst other related to the choice of representative velocity, where Wellens et al. (2010) choose the pore velocity, while e.g. Jensen et al. (2014) has given the momentum equation in filter velocities. Furthermore, there are differences concerning the scaling with the porosity on the various terms in the momentum equation.

waves2Foam was validated for the dambreak through a permeable block in Jensen et al. (2014) and they also validated the overtopping rates against experimental data. Both comparisons showed a good predictive skill of the numerical model. Subsequent works by Jacobsen et al. (2015); Van Gent et al. (2015); Jacobsen et al. (In review) validated waves2Foam against additional experimental data. Several aspects were covered and validated successfully: (i) Wave reflection from hard and porous structures. (ii) Wave damping within a rubble-mound breakwater. (iii) Pressure distribution at the sand-rock interface in an open filter. (iv) Loads on a crest wall element on top of a rubble mound breakwater.

In the work by Jacobsen et al. (2015), an analytical model for the wave-induced setup within a permeable core was derived. The analytical result – that the wave-induced setup is a function of the reflection coefficient – was confirmed with waves2Foam. waves2Foam was subsequently used to developed a numerically based empirical formula for the wave-induced setup outside the validity range of the analytical model.

Finally, waves2Foam was used to predict the effect of ventilation of a bridge deck element with girders against the experimental data by Seiffert et al. (2015) (Jacobsen et al., In review). The considerable effect of even small ventilation holes as observed in the experiments was predicted by the numerical model.

7.3 Breaking waves on a beach profile

Contributor: Niels Gjøl Jacobsen (Deltares). May 2017

A popular study is that of breaking waves on a straight or barred cross-shore profile. The first study with waves2Foam was reported in Jacobsen (2011); Jacobsen et al. (2012) with validation against the experimental data by Ting and Kirby (1994) (spilling breaker). They used a $k-\omega$ turbulence closure with a modified production term following Mayer and Madsen (2000). The comparison is fair, but the dissipation in the inner surfzone was too rapid. The studies furthermore found a strong dependency of the breakpoint with the aspect ratio of the computational cells.

The physical experiments by Ting and Kirby (1994) were later studied by Brown et al. (2014, 2016) (spilling and plunging cases), who considered a multiple of turbulence closures and the influence of the gravity in the turbulence formulation. They saw large differences in the results as a function of the chosen turbulence model and they found it important to include density in the turbulence equations. A ranking of the turbulence closures is provided in Brown et al. (2016) and they recommend the nonlinear $k-\epsilon$ closure.

The measurements of wave height, mean flow and turbulence properties over a fixed concrete bar (Scott, 2005; Scott et al., 2005a,b) were used for validation in Jacobsen et al. (2014); Zhou et al. (2017). While Jacobsen et al. (2014) used a $k-\omega$ model as turbulence closure and performed the simulations in 2DV, Zhou et al. (2017) performed the simulations in 3D with an LES turbulence closure. The results by Jacobsen et al. (2014); Zhou et al. (2017) were compared directly in Zhou et al. (2017) and the differences were not large. Zhou et al. (2017) found that the use of a 3D-model does not solve the problem with excessive turbulence levels, which seems to be a recurring problem for this type of simulations (Bradford, 2000; Christensen, 2006; Brown et al., 2016; Fernandez-Mora et al., 2016).

Finally, it is worthwhile to mention the validation by Fernandez-Mora et al. (2016), who validated the surface elevation signal, mean flow, boundary layer processes, turbulence levels and sediment transport rates against recent experimental work (Van der Zanden et al., 2016, 2017). The general picture confirms the maturity of this type of numerical modelling, though the sediment transport still deserves attention. It should be mentioned, however, that the experimental data is for an evolving morphology, why the evolution of the bathymetry could explain some of the discrepancies between the numerical and experimental results. The importance of the evolving morphology could be addressed by using the experimental data over a frozen bathymetry as reported in Van der A et al. (2017). This is yet to be seen in the literature.

7.4 Modelling of floating wave energy converters (WECs)

Contributor: Edward Ransley (University of Plymouth). August 2017

Despite an R&D process reliant on computationally-efficient numerical tools; the higher-order interactions between the waves and the body, the coupled nature of complete wave energy converter (WEC) systems and, the 'extreme' environment at desirable offshore development sites; dictates that high-fidelity numerical methods are needed in the development of floating WECs.

Davidson et al. (2017) summarises the use of OpenFOAM in wave energy applications including examples in which OpenFOAM simulations of fixed and oscillating WECs have been validated against experimental data. In terms of floating WECs: Devolder et al. (2016) used OpenFOAM to simulate the heave of a floating point-absorber and found good agreement with experimental data for the motion and radiated wave fields in both decay and regular wave tests; Palm et al. (2016) presented a method for coupled mooring analysis combining an OpenFOAM/waves2Foam numerical wave tank (NWT) with an in-house mooring code, MooDy. They show the coupled model is able to capture the non-linear wave height dependence in the response of a moored floating vertical cylinder and, for the surge and heave motion, the numerical results compare well with experimental measurements. For the pitch motion, however, Palm et al. (2016) found the response was under-predicted at resonance but attribute this to uncertainties and simplifications in the buoy properties and geometry; Rafiee and Fiévez (2015) used an OpenFOAM/waves2Foam-based NWT to study wave interaction with a submerged point-absorber. They show that, in moderate regular waves, the Navier-Stokes solver is noticeably more accurate than a linear time-domain model when compared with physical data; Ransley (2015) constructed a NWT, using native OpenFOAM/waves2Foam code, to assess the validity of such methods when analysing the survival characteristics of WECs. Utilising physical measurement data from the experimental campaigns of Jakobsen et al. (2014), Hann et al. (2015) and Göteman et al. (2015), he showed the NWT was capable of reproducing the loading and pressure distribution on fixed WECs as well as the large amplitude motions of constrained devices in large regular waves (Ransley et al., 2017b). Ransley (2015) added a linear generator model as a restraint to the `sixDoFRigidBodyMotion` solver and showed that the NWT was capable of reproducing the complex behaviour of a point-absorbing WEC that undergoes full submersion and breach as well as the load in the mooring caused by the generator's limited stroke length. Sjökvist et al. (2017) used a similar method and showed that, for the same point-absorber, the NWT could predict the motion of the buoy and the load on the generator throughout a focused wave event that had been embedded in a regular wave background. Ransley (2015) showed the NWT was capable of reproducing a âĂŸdesignâĂŹ wave event known as NewWave, correctly predicting the combination of dispersive focusing and energy exchange between frequency components. He went on to show that the NWT could successfully predict the forces and run-up on fixed generic WEC geometries (Ransley, 2015) as well as the motion and mooring loads of simple WEC systems when subject to design waves (Ransley, 2015; Ransley et al., 2017a).

7.5 Wave-structure-seabed interaction

Contributor: Hisham Elsafti (Technical University of Braunschweig). July 2017

The waves2Foam toolbox was applied to study wave-structure-seabed interaction of caisson breakwaters by Elsafti and Oumeraci (2017). The toolbox was semi-coupled in a one-way fashion with the *geotechFoam* solver (Elsafti and Oumeraci, 2016) via the *data2TimeVaryingMappedPressure* boundary condition. The *geotechFoam* solver can simulate fully dynamic, coupled poro-elstoplastic seabed with frictional contact between the structure and the underlying soil. The developed model system was applied to reproduce the large-scale caisson breakwater model tests from the EU research project LIMAS (LIquefaction around MArine Structures) as described fully in Kudella et al. (2006). The setup of the large-scale model tests contained wave gauges, pressure transducers, pore pressure transducers, displacement meters and soil stress transducers. A comparison of the breaking wave pressure on the front face of the caisson breakwater (produced with waves2Foam) and the large-scale tests may be found in Elsafti and Oumeraci (2017). Furthermore, the vertical displacement of the front and back sides of the caisson breakwater subject to breaking wave impact as calculated by *geotechFoam* (coupled with waves2Foam) using the fully dynamic Biot formulation and the $u-p$ approximation against the results of the large-scale tests are also found in Elsafti and Oumeraci (2017).

Bibliography

Arrighi, C., Alcèrraca-Huerta, J. C., Oumeraci, H., Castelli, F., 2015. Drag and lift contribution to the incipient motion of partly submerged flooded vehicles. Journal of Fluids and Structures 57, 170–184.

Beji, S., Battjes, J., 1993. Experimental investigation of wave-propagation over a bar. Coastal Engineering 19 (1-2), 151–162.

Berberović, E., Van Hinsberg, N. P., Jakirlić, S., Roisman, I. V., Tropea, C., 2009. Drop impact onto a liquid layer of finite thickness: Dynamics of the cavity evolution. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 79 (3), Art.no: 036306.

Bradford, S. F., 2000. Numerical simulation of surf zone dynamics. Journal of Waterway, Port, Coastal, and Ocean Engineering 126 (1), 1–13.

Bricker, J. D., Nakayama, A., 2014. Contribution of trapped air, deck superelevation, and nearby structures to bridge deck failure during a tsunami. Journal of Hydraulic Engineering 140 (5), 05014002.

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

Brown, S. A., Magar, V., Greaves, D. M., Conley, D. C., 2014. An evaluation of RANS turbulence closure models for spilling breakers. Proceeding to Coastal Engineering Conference, 1–15.

Chapalain, G., Cointe, R., Temperville, A., 1992. Observed and Modeled Resonantly Interacting Progressive Water-Waves. Coastal Engineering 16 (3), 267–300.

Chappelear, J. E., 1962. Shallow-water waves. Journal of Geophysical Research 67 (12), 4693–4704.

Christensen, E. D., 2006. Large eddy simulation of spilling and plunging breakers. Coastal Engineering 53 (5-6), 463–485.

Davidson, J., Windt, C., Giorgi, G., Genest, R., Ringwood, J. V., 2017. Evaluation of energy maximising control systems for wave energy converters using OpenFOAM. In: Nóbrega, J. M., Jasak, H. (Eds.), OpenFOAM®: Selected papers of the 11th Workshop. Springer International Publishing AG.

Dean, R. G., Dalrymple, R. A., 1991. Water Wave Mechanics for Engineers and Scientists, 1st Edition. Vol. 2 of Advanced Series on Ocean Engineering. World Scientific.

del Jesus, M., Lara, J. L., Losada, I. J., 2012. Three-dimensional interaction of waves and porous coastal structures. Part I: Numerical model formulation. Coastal Engineering , 56–72.

Devolder, B., Rauwoens, P., Troch, P., 2016. Numerical simulation of wave-induced roll of a 2-D rectangular barge using OpenFOAM. In: Proceedings of the 2nd International Conference on Renewable Energies Offshore, RENEW. Lisbo, Portugal.

Devolder, B., Rauwoens, P., Troch, P., 2017. Application of a buoyancy-modified $k-\omega$ SST turbulence model to simulation wave run-up around a monopile subjected to regular waves using OpenFOAM®. Coastal Engineering 125, 81–94.

Dimakopoulos, A. S., Cuomo, G., Chandler, I., 2016. Optimized generation and absorption for three-dimensional numerical wave and current facilities. Journal of Waterway, Port, Coastal and Ocean Engineering 142 (4), 1–13.

Douglass, S. L., Chen, Q., Olsen, J. M., Edge, B. L., Brown, D., 2006. Wave forces on bridge decks. Tech. rep., Office of Bridge Technology, Washington, DC.

Elsafti, H., Oumeraci, H., 2013. Modelling sand foundation behaviour underneath caisson breakwaters subject to breaking wave impact. In: Proc. of the ASME 32nd 2013 Int. Conf. on Ocean, Offshore and Arctic Eng. Nantes, France.

Elsafti, H., Oumeraci, H., 2016. A numerical hydro-geotechnical model for marine gravity structures. Computers and Geotechnics 79, 105–129.

Elsafti, H., Oumeraci, H., 2017. Analysis and classification of stepwise failure of monolithic breakwaters. Coastal Engineering 121, 221–239.

Engelund, F., 1953. On the laminar and turbulent flows of ground water through homogeneous sand. Vol. 3 of Transactions of the Danish Academy of Technical Sciences. Danish Academy of Technical Sciences.

Engsig-Karup, A., August 2006. Unstructured Nodal DG-FEM Solution of High-Order Boussinesq-Type Equations. Ph.D. thesis, Technical University of Denmark.

Engsig-Karup, A. P., Bingham, H. B., Lindberg, O., 2009. An efficient flexible-order model for 3D nonlinear water waves. Journal of Computational Physics 228 (6), 2100–2118.

Ertekin, R. C., Becker, J. M., 1998. Nonlinear Diffraction of Waves by a Submerged Shelf in Shallow Water. J. Offshore Mech. Arct. Eng. 120, pp. 212–220.

Ertekin, R. C., Wehausen, J. V., 1986. Some Soliton Calculations. In: Webster, W. (Ed.), Proc. 16th Symp. on Naval Hydrodynamics, Berkeley. National Academy Press, pp. 167–184.

Fenton, J. D., 1985. A 5th-order Stokes theory for steady waves. Journal of Waterway, Port, Coastal and Ocean Engineering 111 (2), 216–234.

Fenton, J. D., 1988. The Numerical-Solution of Steady Water-Wave Problems. Computers & Geosciences 14 (3), 357–368.

Fenton, J. D., 1990. Nonlinear Wave Theories. Wiley, Ch. 1, pp. 3–25, from *The Sea. Ideas and Observations on Progress in the Study of the Seas. Part A*. Editors: B. Le Méhauté and D. M. Hanes.

Fernandez-Mora, A., Ribberink, J. S., Van der Zanden, J., Van der Werf, J. J., Jacobsen, N. G., 2016. RANS-VOF modelling of hydrodynamics and sand transport under full-scale non-breaking and breaking waves. Proceeding to Coastal Engineering Conference, 1–15.

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

Göteman, M., Engström, J., Eriksson, M., Hann, M., Ransley, E., Greaves, D., Leijon, M., 2015. Scale tests on the survivability of point-absorbing wave energy converters - part i. Journal of Ocean and Wind Energy 2, 176–181.

Guanche, R., Losada, I. J., Lara, J. L., 2009. Numerical analysis of wave loads for coastal structure stability. Coastal Engineering 56, 543–558.

Hann, M., Greaves, D., Raby, A., 2015. Snatch loading of a single taut moored floating wave energy converter due to focussed wave groups. Ocean Engineering 96, 258–271.

Hayatdavoodi, M., Ertekin, R. C., Robertson, I. N., Riggs, H. R., 2015a. Vulnerability assessment of coastal bridges on oahu impacted by storm surge and waves. Natural Hazards 79 (2), 1133–1157.

Hayatdavoodi, M., Seiffert, B., Ertekin, R. C., 2014. Experiments and computations of solitary-wave forces on a coastal-bridge deck. Part II: Deck with girders. Coastal Engineering 88, 210–228.

Hayatdavoodi, M., Seiffert, B., Ertekin, R. C., 2015b. Experiments and calculations of cnoidal wave loads on a flat plate in shallow-water. Journal of Ocean Engineering and Marine Energy 1 (1), 77–99.

Higuera, P., 2015. Application of computational fluid dynamics to wave action on structures. Ph.D. thesis, Universidad de Cantabria.

Higuera, P., Lara, J. L., Losada, I. J., 2014. Three-dimensional interaction of waves and porous coastal structures using OpenFoam®. Part I: Formulation and validation. Coastal Engineering 83, 243–258.

Higuera, P., Losada, I. J., Lara, J. L., 2015. Three-dimensional numerical wave generation with moving boundaries. Coastal Engineering 101, 35–47.

Holthuijsen, L. H., 2007. Waves in Oceanic and Coastal Waters, 1st Edition. Cambridge University Press.

Jacobsen, N. G., 2011. A full hydro- and morphodynamic description of breaker bar development. Ph.D. thesis, Department of Mechanical Engineering, Technical University of Denmark, http://orbit.dtu.dk/recid/313371.

Jacobsen, N. G., Borsboom, M., Van Gent, M. R. A., In preparation. Investigation of the KC-dependency in the extended Darcy-Forchheimer resistance formulation. – (–), –.

Jacobsen, N. G., Fredsøe, J., 2014a. Cross-Shore Redistribution of Nourished Sand Near a Breaker Bar. Journal of Waterway, Port, Coastal and Ocean Engineering 140 (2), 125–134.

Jacobsen, N. G., Fredsøe, J., 2014b. Formation and Development of a Breaker Bar under Regular Waves. Part 2: Sediment Transport and Morphology. Coastal Engineering 88, 55–68.

Jacobsen, N. G., Fredsøe, J., Jensen, J. H., 2014. Formation and Development of a Breaker Bar under Regular Waves. Part 1: Model Description and Hydrodynamics. Coastal Engineering 88, 182–193.

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

Jacobsen, N. G., Van Gent, M. R. A., Capel, A., Borsboom, M., In review. Numerical prediction of wave loads on crest walls on top of rubble mound structures. Coastal Engineering –, –.

Jacobsen, N. G., Van Gent, M. R. A., Wolters, G., 2015. Numerical analysis of the interaction of irregular waves with two dimensional permeable coastal structures. Coastal Engineering 102, 13–29.

Jakobsen, M. M., Iglesias, G., Kramer, M., Vidal, E., 2014. Experimental study of forces on point absorber. In: Proceedings of the 5th International Conference Coastlab 14. Varna, Bulgaria.

Jensen, B., Jacobsen, N. G., Christensen, E. D., 2014. Investigations on the porous media equations and resistance coefficients for coastal structures. Coastal Engineering 84, 56–72.

Kudella, M., Oumeraci, H., De Groot, M., Meijers, P., 2006. Large-scale experiments on pore pressure generation underneath a caisson breakwater. Journal of Waterway, Port, Coastal, and Ocean Engineering 132 (4), 310–324.

Kulicki, J. M., Mertz, D. R., 2008. Guide Specifications for Bridges Vulnerable to Coastal Storms. AASHTO.

Liu, P. L. F., Lin, P., Chang, K. A., Sakakiyama, T., 1999. Numerical modeling of wave interaction with porous structures. Journal of Waterway, Port, Coastal and Ocean Engineering 125, 322–330.

Losada, I. J., Lara, J. L., Del Jesus, M., 2016. Modeling the interaction of water waves with porous coastal structures. Journal of Waterway, Port, Coastal and Ocean Engineering 142 (6), 1–18.

Losada, I. J., Lara, J. L., Guanche, R., Conzalez-Ondina, J. M., 2008. Numerical analysis of wave overtopping of rubble mound breakwaters. Coastal Engineering 55, 47–62.

Madsen, P. A., Fuhrman, D. R., 2006. Third-order theory for bichromatic bi-directional water waves. Journal of Fluid Mechanics 557, 369–397.

Madsen, P. A., Fuhrman, D. R., 2012. Third-order theory for multi-directional irregular waves. Journal of Fluid Mechanics 698, 304–334.

Mayer, S., Madsen, P. A., 2000. Simulation of Breaking Waves in the Surf Zone using a Navier-Stokes Solver. Proceeding to Coastal Engineering Conference I, 928–941.

McPherson, R. L., 2008. Hurricane Induced Wave and Surge Forces on Bridge Decks. Ph.D. thesis, Texas A&M University.

Moradi, N., 2015. Numerical simulation of fluid resonance in the narrow gap of twin bodies in close proximity. Ph.D. thesis, School of Civil, Environmental and Mining Engineering, The University of Western Australia.

Palm, J., Eskilsson, C., Paredes, G. M., Bergdahl, L., 2016. Coupled mooring analysis for floating wave energy converters using CFD: formulation and validation. International Journal of Marine Energy 16, 83–99.

Patarapanich, M., 1984. Forces and moment on a horizontal plate due to wave scattering. Coastal Engineering 8 (3), 279–301.

Paulsen, B. T., Bredmose, H., Bingham, H. B., 2014a. An efficient domain decomposition strategy for wave loads on surface piercing circular cylinders. Coastal Engineering 86, 57–76.

Paulsen, B. T., Bredmose, H., Bingham, H. B., Jacobsen, N. G., 2014b. Forcing of a bottom-mounted circular cylinder by steep regular water waves at finite depth. Journal of Fluid Mechanics 755, 1–34.

Rafiee, A., Fiévez, J., 2015. Numerical prediction of extreme loads on the ceto wave energy converter. In: Proceedings of the 11th European Wave and Tidal Energy Conference (EWTEC). Nantes, france.

Ransley, E., 2015. Survivability of wave energy converter and mooring coupled system using cfd. Ph.D. thesis, Plymouth University.

Ransley, E., Greaves, D., Raby, A., Simmonds, D., Hann, M., 2017a. Survivability of wave energy converters using CFD. Renewavle Energy 109, 235–247.

Ransley, E., Greaves, D., Raby, A., Simmonds, D., Jakobsen, M., Kramer, M., 2017b. RANS-VOF modelling of the wavestar point absorber. Renewavle Energy 109, 49–65.

Rienecker, M. M., Fenton, J. D., 1981. A Fourier Approximation Method for Steady Water-Waves. Journal of Fluid Mechanics 104, 119–137.

Rusche, H., December 2002. Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions. Ph.D. thesis, Imperical College of Science, Technology and Medicine, available at: http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/HenrikRuschePhD2002.pdf.

Scott, C. P., 2005. Large-Scale Laboratory Observations of Wave Breaking Turbulence on a Fixed Barred Beach. Master thesis.

Scott, C. P., Cox, D. T., Maddux, T. B., Long, J. W., 2005a. Large-scale laboratory observations of turbulence on a fixed barred beach. Measurement, Science & Technology 16 (10), 1903–1912.

Scott, C. P., Cox, D. T., Shin, S., Clayton, N., 2005b. Estimates of surf zone turbulence in a largescale laboratory flume. Proceeding to Coastal Engineering Conference, 379–391.

Seiffert, B., Ertekin, R. C., Robertson, I. N., 2015. Wave loads on a coastal bridge deck and the role of entrapped air. Applied Ocean Research 53, 91–106.

Seiffert, B., Hayatdavoodi, M., Ertekin, R. C., 2014. Experiments and computations of solitary-wave forces on a coastal-bridge deck. Part I: Flat Plate. Coastal Engineering 88, 194–209.

Seiffert, B. R., 2014. Tsunami and storm wave impacts on coastal bridges. Ph.D. thesis, University of Hawaii.

Seiffert, B. R., Hayatdavoodi, M., Ertekin, R. C., 2015. Experiments and calculations of cnoidal wave loads on a coastal-bridge deck with girders. European Journal of Mechanics - B/Fluids 52, 191–205.

Seng, S., 2012. Slamming and whipping analysis of ships. Ph.D. thesis, Department of Mechanical Engineering, Technical University of Denmark.

Sharma, J. N., Dean, R. G., 1981. 2nd-Order Directional Seas and Associated Wave-Forces. Society of Petroleum Engineers Journal 21 (1), 129–140.

Sjökvist, L., Wu, J., Ransley, E., Engström, J., Eriksson, M., Göteman, M., 2017. Numerical models for the motion and forces of point-absorbing wave energy converters in non-linear waves. Ocean EngineeringManuscript in review.

Stahlmann, A., 2013. Experimental and Numerical Modeling of Scour at Offshore Wind Turbines. Ph.D. thesis, Gottfried Wilhelm Leibniz Universität Hannover.

Sun, X., 1991. Some theoretical and numerical studies on cnoidal-wave-diffraction problems. Ph.D. thesis, MasterâĂŹs thesis, University of Hawaii at Manoa, Honolulu.

Svendsen, I. A., Jonsson, I. G., 1982. Hydrodynamics of Coastal Regions, 1st Edition. Den Private Ingeniørfond.

Ting, F. C. K., Kirby, J. T., 1994. Observation of Undertow and Turbulence in a Laboratory Surf Zone. Coastal Engineering 24 (1-2), 51–80.

Van der A, D. A., Van der Zanden, J., O'Donoghue, T., Hurther, D., Cáceres, I., McLelland, S. J., Ribberink, J. S., 2017. Large-scale laboratory study of breaking wave hydrodynamics over a fixed bar. Journal of Geophysical Research – Oceans N/A, N/A.

Van der Zanden, J., Van der A, D. A., Hurther, D., Cáceres, I., O'Donoghue, T., Ribberink, J. S., 2016. Near-bed hydrodynamics and turbulence below a large-scale plunging breaking wave over a mobile barred bed profile. Journal of Geophysical Research – Oceans 121 (8), 6482–6506.

Van der Zanden, J., Van der A, D. A., Hurther, D., Cáceres, I., O'Donoghue, T., Ribberink, J. S., 2017. Suspended sediment transport around a large-scale laboratory breaker bar. Coastal Engineering 125, 51–69.

Van Gent, M. R. A., 1995. Porous flow through rubble mound material. Journal of Waterway, Port, Coastal and Ocean Engineering 121 (3), 176–181.

Van Gent, M. R. A., Herrera, M. P., Molines, J., Jacobsen, N. G., 2015. Rock slopes on top of sand: Modelling of open filters under wave loading. Proceeding to Coastal Structures, 1–12.

Van Gent, M. R. A., Tönjes, P., Petit, H. A. H., Van den Bosch, P., 1994. Wave action on and in permeable structures. Proceeding to Coastal Engineering Conference, 1739–1753.

Vukcević, V., Jasak, H., Malenica, S., 2016a. Decomposition model for naval hydrodynamic applications, Part I: Computational method. Ocean Engineering 121, 37–46.

Vukcević, V., Jasak, H., Malenica, S., 2016b. Decomposition model for naval hydrodynamic applications, Part II: Verification and validation. Ocean Engineering 121, 76–88.

Wellens, P. R., January 2012. Wave Simulation in Truncated Domains for Offshore Applications. Ph.D. thesis, Technical University of Delft.

Wellens, P. R., Borsboom, M. J. A., Van Gent, M. R. A., 2010. 3D simulation of wave interaction with permeable structures. Proceeding to Coastal Engineering Conference, 1–15.

Zelt, J. A., Skjelbreia, J. A., 1992. Estimating Incident and Reflected Wave Fields Using an Arbitrary Number of Wave Gauges. Proceeding to Coastal Engineering Conference I, 777–789.

Zhou, Z., Hsu, T.-J., Cox, D., Liu, X., 2017. Large-eddy simulation of wave-breaking induced turbulent coherent structures and suspended sediment transport on a barred beach. Journal of Geophysical Research – Ocean 122 (2016JC011884).

APPENDIX A

Overview of Input Files

There is a number of files necessary to make a successful setup of a wave simulation with waves2Foam. Some of these files are native to OpenFoam, while others are additional input files (pre- and postprocessing and run-time). All files are briefly discussed in this chapter to give the user an overview of the origin and meaning of the files.

Roughly speaking, the files placed in the `system`-folder are related to the time control and numerical schemes of the simulation, while the files placed in the `constant`-folder are related to the physical properties of the simulation.

The following files are native to OpenFoam:

- `system`-folder:
  - `controlDict`
  - `fvSchemes`
  - `fvSolution`
- `constant`-folder:
  - `dynamicMeshDict`
  - `g`
  - `transportProperties`
  - `turbulenceProperties`
  - `RASProperties`
  - `LESProperties`
  - `porosityZones`

The following files are additional files native to waves2Foam:

- `system`-folder:
  - None
- `constant`-folder:
  - `waveProperties.input`
  - `waveProperties`
  - `probeDefinitions`
  - `postProcessingProperties`
  - `wavesPorosityProperties`
  - `triSurface/stlDefinitions`

In section A.1 files native to OpenFoam are described, while files native to waves2Foam are described in section A.2

A.1 Files native to OpenFoam

A.1.1 `system/controlDict`

The `controlDict` controls the duration of the simulation, the output interval and the adaptive time step (if activated). This is also the file, where function objects are activated in the simulation. Function objects allow for run-time evaluation of additional properties: The total force on a structural element, the surface elevation, or the pressure in specific locations with the probe tool.

A.1.2 system/fvSchemes

`fvSchemes` controls the numerical schemes used to discretise the system. This relates to both the temporal and spatial discretisations.

A.1.3 system/fvSolution

`fvSolution` controls the solution procedure for the linear systems of equation. This includes the specification of solution tolerances, linear solvers, under-relaxation and the number of iterations in the pressure-velocity coupling.

A.1.4 constant/dynamicMeshDict

The `dynamicMeshDict` controls mesh motion, dynamic mesh refinement or topological changes. The keywords in this file depends on the flavour of the OpenFoam-branch.

A.1.5 constant/g

This files is used to define the direction and magnitude of the acceleration due to gravity.

A.1.6 constant/transportProperties

`transportProperties` is used to define the viscosity and density of the fluids. Also, the surface tension coefficent between two fluids is defined in this file. In the case of free surface waves, the surface tension is typically neglected.

A.1.7 constant/turbulenceProperties

`turbulenceProperties` controls, whether a laminar, RANS or LES turbulence closure is applied in the simulation; see section 2.4 for more details on turbulence modelling and waves.

A.1.8 constant/RASProperties

If a RANS-type turbulence closure is used, the actual turbulence closure is specified in `RASProperties`.

A.1.9 constant/LESProperties

If a LES-type turbulence closure is used, the actual turbulence closure is specified in `LESProperties`.

A.1.10 constant/porosityZones

The resistance properties of porous zones are specified in `porosityZones`.

A.2 Files related to waves2Foam

A.2.1 constant/waveProperties.input

`waveProperties.input` is the main input file for wave modelling. Here, the specifications are given in terms of wave theory, environmental properties, etc.

A.2.2 constant/waveProperties

`waveProperties` is a result of pre-processing of `waveProperties.input`. It can also be generated by hand, but it is generally discouraged.

A.2.3 constant/probeDefinitions

`probeDefinitions` is used to specify the locations of wave gauges and probes.

A.2.4 constant/postProcessingProperties

`postProcessingProperties` is used to specify post-processing steps, e.g. zero-crossing analysis of the evaluation of the reflection of a regular wave.

A.2.5 constant/triSurface/stlDefinitions

This file is used to define simple STL-surfaces. STL-surfaces may be used for the meshing with `snappyHexMesh` or for the definition of porous zones.

APPENDIX B

Treatment of Irregular Waves

This appendix discusses the impact of two different methods to discretise the frequency axis for a wave spectrum. In addition to this, the appendix describes the `irregularFast` approach to evaluate an irregular wave signal (Section 4.3.2). The combination of these two components offers a way to accelerate the evaluation of irregular incident waves; the method differs from the work by Dimakopoulos et al. (2016), who applied Taylor expansions of the trigonometric functions.

The content of this work is realised in collaboration with Bo Terp Paulsen. The content has not been published elsewhere. The style of the appendix has the flavour of a journal contribution as this was the original ambition, however, the lack of time prevented the finalisation of the manuscript, why it is adapted to the manual for future reference.

B.1 Introduction

In the field of coastal engineering it is common practice to evaluate the design of coastal structures by means of physical experiments and numerical modelling. For the representation of a natural sea state, irregular waves are described by a spectral distribution of the wave energy. Several wave spectra based on sea state measurements have become standard practice both for physical experiments and for numerical modelling, e.g., the JONSWAP spectrum and the Pierzon-Moscowitch spectrum.

For numerical models, the volume-of-fluid (VOF) type models are often applied to track the free water surface in combination with Navier-Stokes models. Also Boussinesq type models and non-linear potential wave models are applied when a numerical approach is taken towards the evaluation of wave-structure interaction.

In order to generate an irregular sea state based on a wave spectrum, both experimentally and numerically, the incident wave spectrum must be discretised into a number of discrete frequencies. Each discrete frequency represent a regular wave component, and the superposition of all discrete regular wave components forms the irregular sea state.

In physical laboratory facilities the spectral discretisation is typically chosen as $\Delta f = 1/T_e$, where f is the frequency and T_e is the duration of the experiment. Hereby an equidistant discretisation is achieved which adds the same number of regular wave components both at the part of the wave spectrum that contains a relatively high amount of energy and a low amount of energy.

This approach can also be used for the numerical wave modelling, however, when the combined wave generation and reflection compensation is achieved with the use of relaxation zones (see e.g. Jacobsen et al., 2012) it become computationally expensive to evaluate thousands of wave components at every time step. For practical applications a limited number of discrete wave components are usually applied, e.g., in the order of 100 components. Therefore it becomes important how the discrete components are distributed along the frequency range. As will be shown, both the discretisation and the total number of wave components affects the return period of the signal and the wave height distribution.

First, some mathematical considerations are presented with respect to the return period of the time series as evaluated from a spectrum with a given frequency discretisation. Secondly, the choice on frequency discretisation and resolution is studied with an emphasis on the integrated spectral properties and the exceedance distribution of the wave heights and wave periods. Finally, an alternative approach to evaluate the irregular wave signal is presented, which accelerates the execution of the solvers in waves2Foam.

B.2 Mathematical description

A set of generated irregular incident waves must avoid a repetition of the signal within the modelled duration of the experiment (physical or numerical). The period of repetition of the signal can be evaluated

by a calculation of the autocorrelation; either using the time series of the surface elevation or by applying the Wiener-Khinchin theorem on the real-valued spectrum:

$$R(\tau) = \frac{1}{m_0}\int_0^{-\infty} S(f)\cos 2\pi f\tau \mathrm{d}f \tag{B.1}$$

Here $R(\tau)$ is the autocorrelation function, $S(f)$ is the spectrum, f is the frequency, τ is the correlation time and m_0 is the zero order moment of the spectrum.

The discrete version of Eq. (B.1) is given as follows using the trapezoidal integration rule:

$$R_d(\tau) \simeq \frac{1}{2m_{d,0}}\sum_{n=0}^{N-1}\left[S_d(f_n)\cos(2\pi f_n\tau) + S_d(f_{n+1})\cos(2\pi f_{n+1}\tau)\right]\Delta f_n \tag{B.2}$$

Here, sub-index 'd' refers to the discrete values. The discretisation is such that $\Delta f_n = f_{n+1} - f_n$. N is the number of discrete wave components in the time domain representation of the surface elevation (see below).

The spectral discretisation of the surface elevation η is transformed into the time domain by the following linear super-position:

$$\eta(t,\mathbf{x}) = \sum_{m=1}^{N} a_m \cos\left(\omega_m t - \mathbf{k}_m\cdot\mathbf{x} + \varphi_m\right) \tag{B.3}$$

a_m is the amplitude corresponding to the cyclic frequency ω_m, $\mathbf{k}_m$ the wave number vectors, $\mathbf{x}$ is the Cartesian coordinate vector and φ_m the random phases in the interval $[0,\ 2\pi]$. Here,

$$a_m = \sqrt{[S_d(f_{m-1}) + S_d(f_m)]\Delta f_{m-1}} \tag{B.4}$$

and

$$\omega_m = \pi(f_m + f_{m-1}) \tag{B.5}$$

B.2.1 Repetition of the signal

The signal is repeated, when $\tau > 0$ s and all of the cosines are in phase, i.e. $\cos(2\pi f_n\tau) = 1$ for all values of n. If the period of repetition, T_r, is small relative to the mean wave period, T_{m01}, it means that there can only be a limited number of wave heights in the signal. Consequently, the wave height distribution will deviate considerably from a Rayleigh distribution; especially for small values of the exceedance probability. The effect of the discretisation of the spectrum on T_r is considered here for an equidistant and a simple (linear) non-equidistant discretisation.

B.2.1.1 Equidistant discretisation

The equidistant discretisation is given by $f_n = n\delta f_0$ and $\Delta f_n = \delta f_0$ for $n = 0, 1\ldots, N$. It is easy to obtain the well established relationship

$$T_r = \frac{1}{\delta f_0} = \frac{N}{f_N} \tag{B.6}$$

by inserting the equidistant discretisation into Eq. (B.2). f_N is the upper cut-off frequency. It is seen that the larger N (or smaller f_N), the larger the value of T_r.

B.2.1.2 Non-equidistant discretisation

A simple non-equidistant discretisation is considered in this section, namely

$$\Delta f_n = (1 + n\alpha_f)\delta f_0 \tag{B.7}$$

Here

$$f_n = n\delta f_0 + \alpha_f\delta f_0 n\left[\frac{n+1}{2} - 1\right] = \delta f_0(n + \alpha_f A_n) \tag{B.8}$$

for $1/(1-N) < \alpha_f$ (the equidistant discretisation is re-found for $\alpha_f = 0$). Inserting this discretisation into Eq. (B.2), the cosine terms can be expanded as follows:

$$\begin{aligned}\cos(2\pi f_n\tau) =& \cos(2\pi n\delta f_0\tau)\cos(2\pi\alpha_f A_n\delta f_0\tau)\\ &- \sin(2\pi n\delta f_0\tau)\sin(2\pi\alpha_f A_n\delta f_0\tau)\end{aligned} \tag{B.9}$$

It is observed that the autocorrelation for the equidistant discretisation has been overlayed by a second harmonic, which differs for each value of n. In order to obtain a repetition of the signal it must be required for the cosine terms that

$$n\delta f_0\tau = K \quad \text{and} \quad \alpha_f A_n\delta f_0\tau = L \tag{B.10}$$

at the same time; here the sinusoidal terms are ignored, because they are out of phase with the cosine terms. K and L are integers. Since τ is identical in both equalities, it is found that

$$\frac{K}{n}\frac{L}{A_n}\alpha_f = 1 \tag{B.11}$$

Consequently, if α is not a rational number, there will never be a repetition of the signal, when a repetition is defined as the value of τ for which $R_d = 1$.

The influence of α_f on T_r is exemplified in the following. For $\alpha = 1/4$, $T_r = 1/\delta f_0$, while $T_r = 4/\delta f_0$ for $\alpha = 4$. Note, however, than the value of α_f also affects the magnitude of δf_0, see Eq. (B.8), such that δf_0 decreases with increasing α_f.

This analysis has shown that T_r can be affected strongly by the discrete representation of the spectrum. This attribute is further analysed in the following sections. The simple linear expansion of the frequency axis (Eq. (B.7)) will, however, not be analysed further, because it cannot efficiently represent the peak of the spectrum, which is typically in the middle of the discretised frequency interval.

B.2.2 Discrete representations

In the following, two discrete representations of the spectrum S are analysed with respect to the repetition of the wave signal, the integrated wave properties and the exceedance distribution of the wave heights and wave periods. The discrete representations are described in the following:

B.2.2.1 Equidistant

The equidistant discretisation is already presented aboute in Section B.2.1.1. This discretisation is included in the analysis, because it is the commonly adopted discretisation.

B.2.2.2 Cosine stretching

The cosine stretching is a non-equidistant discretisation of the frequency axis, such that the discretisation is finer adjacent to the peak of the spectrum. This approach is simple to prescribe, but it is difficult to apply on e.g. double-peaked spectra. The method works with a lower, f_L, a peak, f_P and an upper, f_U, frequency.

For the interval $[f_L, f_P[$ the spectrum is discretised with the following number of frequencies:

$$N_L = \frac{f_P - f_L}{f_U - fP}(N+1)\Bigg\rfloor \tag{B.12}$$

The number of frequencies in the interval $[f_P, f_U]$ is consequently $N_U = N + 1 - N_L$. The frequency discretisation is given as follows for the lower part of the spectrum

$$f_n = f_L + (f_P - f_L)\sin\left(\frac{2\pi}{4N_L}n\right) \quad \text{for} \quad n = 0, \ldots N_L - 1 \tag{B.13}$$

while the upper part of the frequency discretisation is given as

$$f_{N_L+n} = f_P + (f_U - f_P)\left[1 - \cos\left(\frac{2\pi}{4N_U}n\right)\right] \quad \text{for} \quad n = 0, \ldots N_U \tag{B.14}$$

B.2.2.3 The discrete representations

The variation in Δf with f for the applied discretisations is depicted in Figure B.1A. Here, $f_L = 0.03$ Hz, $f_U = 1.4$ Hz, $T_p = 3$ s and $N = 200$.

The discrete representation of a JONSWAP spectrum with a peak enhancement factor of 3.3 with $N = 35$ is plotted in Figure B.1B for the two discretisation methods. It is clearly seen that for a small N, the equidistant discretisation cannot capture the peak, whereas the non-equidistant method is tailored to capture the peak. There are no marked visual differences for $N = 200$ between the two discretisation methods.

B.3 Spectral discretisation

B.3.1 Time series and autocorrelation

The time series of the surface elevation and autocorrelation for a spectrum discretised with $N = 20$ are discussed in this section. The remaining parameters are the same as in Figure B.1 and the spectrum is a JONSWAP spectrum with a peak enhancement factor of 3.3. The small value for N was chosen for visual reasons. The surface elevation is depicted for the two discretisations in Figure B.2 and while it is easy to observe a repetition in the signal for the equidistant discretisation (Figure B.2A), there is no obvious repetition of the irregular wave signal for the first 100 s for the non-equidistant discretisation.

The difficulty in observing a repetition of the surface elevation in the signals based on a non-equidistant frequency discretisation was addressed by evaluating the autocorrelation using Eq. B.2. The autocorrelations are depicted in Figure B.3 for the spectral discretisations in Figure B.2. It is appearent that while $T_r \simeq 15$ s for the equidistant discretisation, then T_r is much larger for the non-equidistant discretisation with the same value of N. The value of $|R_d|$ hardly exceeds 0.5 during the first 500 s of the autocorrelation, i.e. an increase in T_r by a factor of 30 was obtained through a simple change in discretisation method. The effect of the discretisation method on the distribution of the individual wave heights and wave periods is presented in Section B.3.3

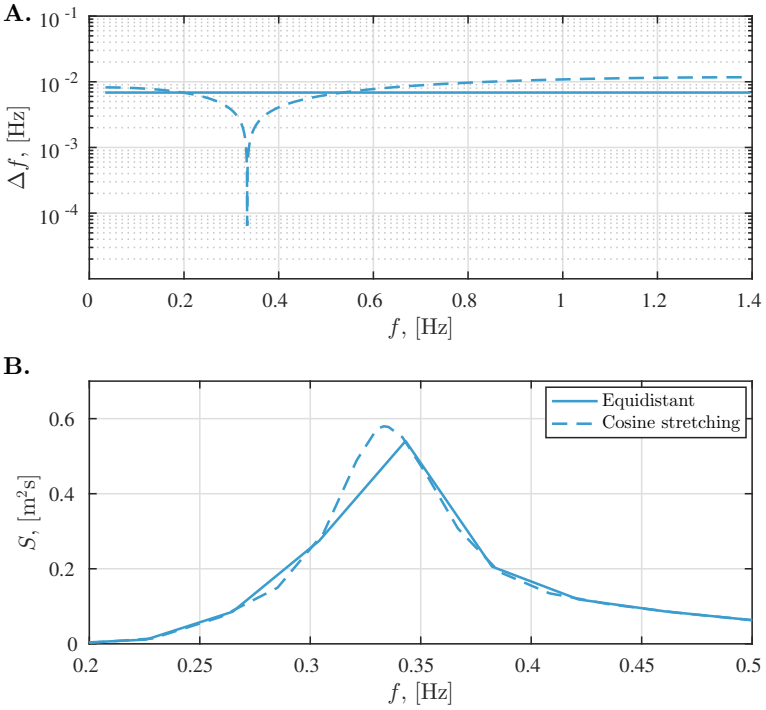

Figure B.1: The variation of Δf as a function of f for the two discretisations of the frequency axis. $f_L = 0.03$ Hz, $f_U = 1.4$ Hz, $T_p = 3$ s. A: $N = 200$. B: $N = 35$ and a JONSWAP spectrum with a peak enhancement factor of 3.3.

B.3.2 Spectral wave properties

The validity of the newly proposed discretisation methods of the spectra requires that the spectral wave properties converge. The convergence of the spectral wave properties is analysed. The evaluated wave properties are the spectral wave height $H_{m0} = 4\sqrt{m0}$ and the three wave periods $T_{m01} = m_0/m_1$, $T_{m02} = \sqrt{m_0/m_2}$ and $T_{m-10} = m_{-1}/m_0$; they are evaluated as a function of the discretisation and N. Here, m_i is the i'th moment of the spectrum and it is defined as

$$m_i = \int_0^\infty f^i S(f) \mathrm{d}f \tag{B.15}$$

Its discrete approximation was evaluated by a trapezoidal integration.

The utilised spectrum was the same as the one defined in conjunction with Figure B.1. A range of N from 10 to 10,000 was used to calculate the spectra and the discrete wave properties. For $N = 10,000$ the spectral wave properties deviated less than 0.1% between the discretisation methods and the spectral wave properties for $N = 10,000$ were adopted as the converged solution.

The results are depicted in Figure B.4 and it is clear that even for N as small as 30, an accuracy within a few percentages was obtained (for this specific choice of spectrum). The noise in the equidistant discretisation was attributed to the discretisation of the peak.

B.3.3 Probability distribution

The integrated spectral wave properties were seen to be insensitive to the spectral discretisation, but what is the impact on the discretisation on the exceedance propability of the wave heights and wave periods?

A time series analysis was conducted on 1,000 realisations of the same spectrum, i.e. 1,000 different sets of random phases. This was done for the two frequency discretisations and four values of N: 100, 200, 500 and 1,000. The surface elevation signals were generated from a linear super-position as in Eq. (B.3). The time series were generated with $\Delta t = 0.02$ s and they had a duration of $T_e = 5,000$ s, which is much longer than T_r for all of the equidistant discretisations ($T_r = 730$ s for $N = 1,000$).

The wave height distribution based on a zero-down crossing is presented in Figure B.5 and it is seen that this distribution is very sensitive to N for the equidistant stretching, whereas the non-equidistant discretisation is less sensitive to N. It is seen that even for as few as $N = 100$, the non-equidistant discretisation is close to the Rayleigh distribution for an exceedance probability of 0.1%, while the equidistant discretisation does not even represent the 1.0% exceedance probability for $N = 100$. There is an uncertainty band around the mean exceedance distribution for all three discretisation methods,

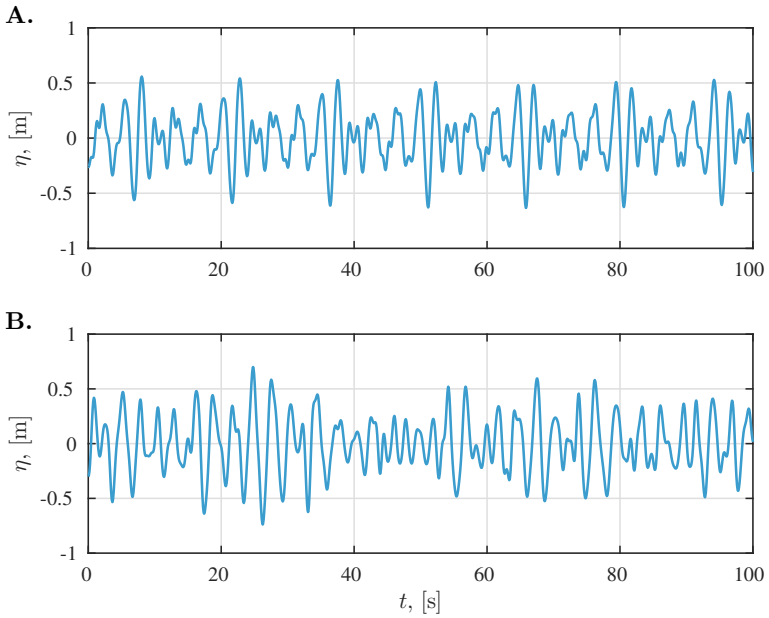

Figure B.2: One realisation of the surface elevation. A: Equidistant discretisation. B: Cosine stretching.

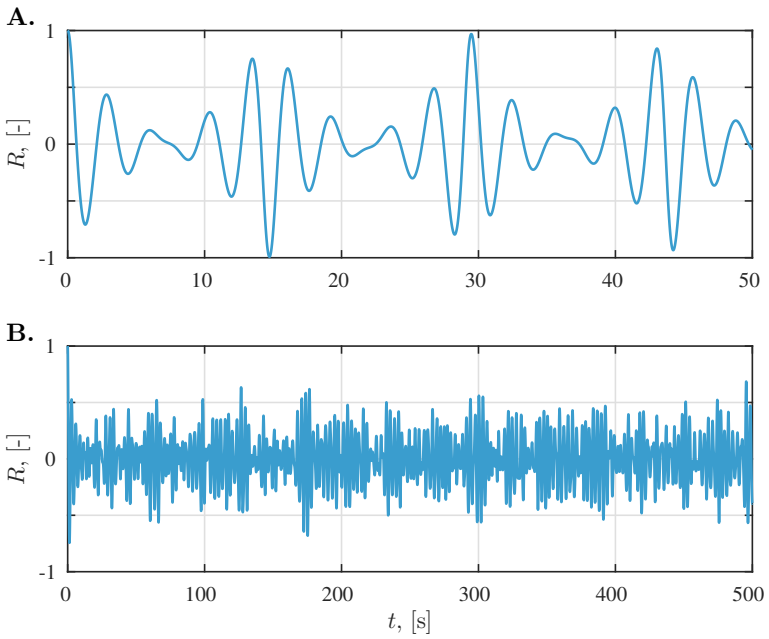

Figure B.3: The discrete autocorrelation of a given spectral discretisation. A: Equidistant discretisation. B: Cosine stretching.

and the uncertainty band is defined by one standard deviation σ. σ is the largest for the equidistant discretisation.

The results from the zero-crossing analysis were also used to evaluate the distribution of the wave periods. The distribution of the wave periods are depicted in Figure B.5 as well. The probability distribution of the wave periods led to the same conclusions, namely that the non-equidistant discretisation produces a time series signal that is less sensitive to the number of wave components.

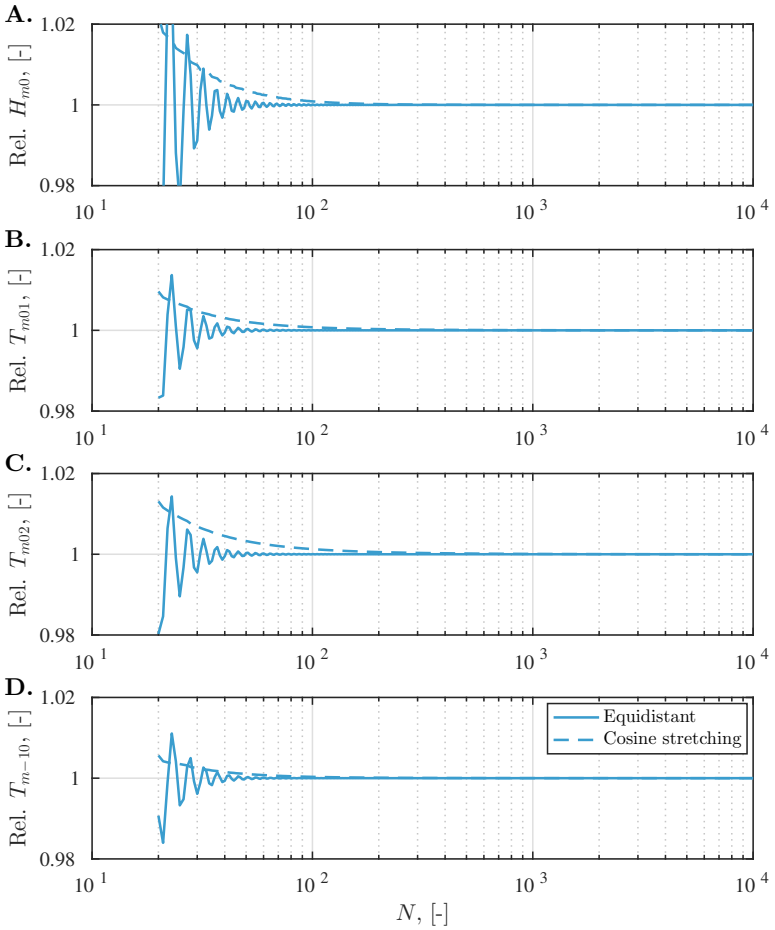

Figure B.4: The variation in the spectral wave properties as a function of N. 'Rel.' means normalisation with the quantity for $N = 10,000$.

B.3.4 Discussion

It appears straightforward that the non-equidistant frequency discretisation has many benefits over the equidistant discretisation. This is true, if the main focus is to produce lightweight time series, which exhibit the correct exceedance properties. However, if the analysed system has resonance frequencies within the wave frequency band, then one should be careful in applying the lightweight discretisation without critical analysis of the results and sensitivity to the number of frequency components.

B.4 Accelerated evaluation of irregular waves

One element of a fast evaluation of an irregular wave signal is the non-equidistant discretisation as discussed above. A second element is a formulation of the problem, where a significant part of the work can be done as a pre-processing step.

First of all, consider the simple superposition of the free surface elevation (direct evaluation):

$$\eta = \eta_0 + \sum_{n=1}^{N} a_n \cos(\omega_n t - \mathbf{k}_n \cdot \mathbf{x} + \phi_n) \tag{B.16}$$

Here, η is the surface elevation, η_0 is the offset from still water level, N is the number of components, ω is the cyclic frequency, t is time, $\mathbf{k}$ is the wave number vector, $\mathbf{x}$ is the coordinate and ϕ is a phase. It is apparent that the sum (Eq. B.16) must be evaluated for every coordinate in the relaxation zone every time step, since the cosine depends on both space and time. It is known that trigonometric function are notoriously expensive to compute.

A simple manipulation of the equation leads to this expression:

$$\eta = \eta_0 + \sum_{n=1}^{N} a_n \left[\cos(\omega_n t + \phi_n) \cos \mathbf{k}_n \cdot \mathbf{x} - \sin(\omega_n t + \phi_n) \sin \mathbf{k}_n \cdot \mathbf{x}\right] \tag{B.17}$$

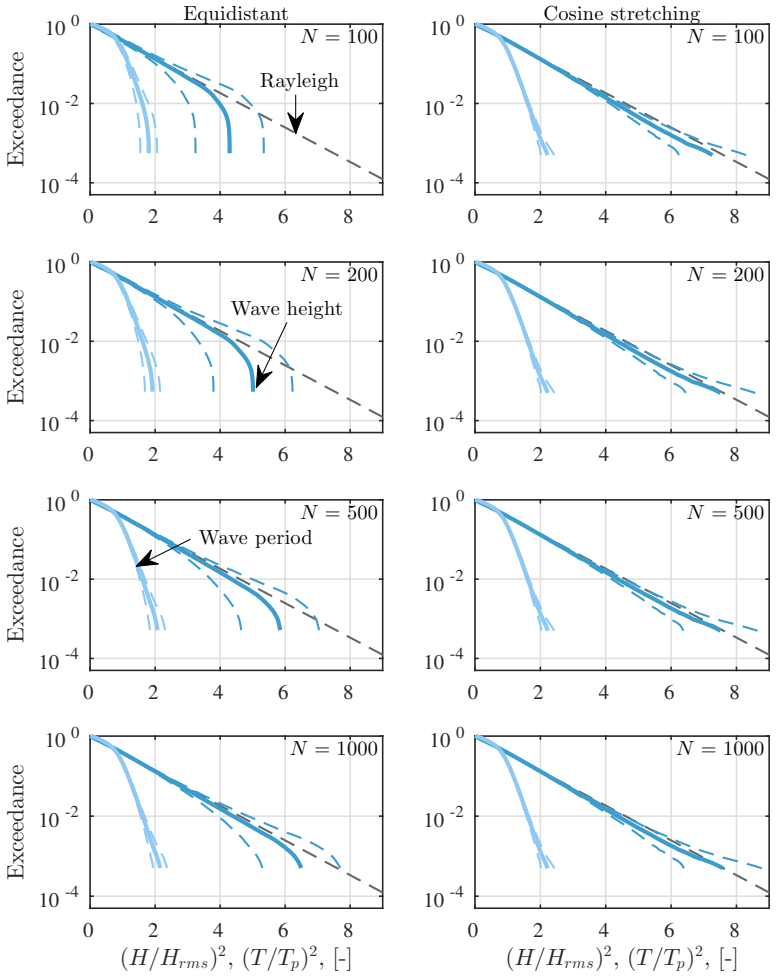

Figure B.5: The probability distribution for both $(H/H_{rms})^2$ and $(T/T_p)^2$ as a function of the discretisation of the frequency axis (columns) and the number of wave components (rows). Full lines is the mean values and dashed lines the mean $\pm\sigma$.

This shows that it is possible to reformulate Eq. (B.16) such that each trigonometric function only depends on space *or* time. Consequently, all trigonometric functions dependent on $\mathbf{x}$ can be evaluated as a post-processing step prior to the time stepping. All trigonometric functions dependent on t must be re-evaluated each time step (before relaxation), but it is cheap, as there are merely $2N$ evaluation, while the number of spatial coordinates is typically much, much larger.

A similar splitting can be performed for the velocity vector, where the vertical coordinate (z) is already split from the remainding arguments: t, x and y.

At first glance, Eq. (B.17) may look more computationally heavy, but since *all* components are pre-computed, the final summation is considerably faster, because multiplication and memory lookup is outperforms evaluation of trigonometric and hyperbolic functions.

This approach is termed the *split method* in the following.

B.4.1 Implementation

The split method in waves2Foam IS termed `irregularFast` (Section 4.3.2). The implementation is based on the external wave theories class (Section 4.3), which allows for the necessary pre-processing steps within a time step.

The implementation is divided into three components, namely (i) global pre-processing step, (ii) local pre-processing step within a time step, and (iii) evaluation of boundary conditions and relaxation zones:

- Global pre-processing
  - Make a list of the unique x, y and z coordinates.
  - Evaluate all trigonometric and hyperbolic functions in these coordinates for all frequqency components and store the values. These lists are effectively matrices with dimensions of the number of coordinates and the number of frequencies.

- Local pre-processing
  - Evaluate the trigonometric functions at the current time instance and store the values. These lists are column vectors with the length of the number of frequencies.
  - Evaluate the surface elevation in all combinations of (x, y) (assume z to be vertical), since the surface elevation is required for the relaxation zone evaluation. This step reduces the computational requirements further.
- Evaluation
  - Look-up the pre-calculated values and perform the necessary multiplications.

The method is only implemented for static grids, but there is a flag to ignore mesh motion, if it is known that the mesh deformation in the relaxation zone is small (Section 4.3.2).

B.4.2 Example

An example of the computational gain is presented in Figure B.6. The execution time was measured for both the direct evaluation of the irregular wave signal (see Eqn. (B.16)) and the proposed split method (see Eqn. (B.17)). Here, the relative execution time was obtained by normalising by the execution time for $N = 1$.

It is seen that there is a steady increase in time consumption for the direct method, while the split method flat-lines up to $N = 100$ (Figure **??**). The flat-lining essentially shows that the costs related to the relaxing method is more expensive that the evaluation of the wave signal itself.

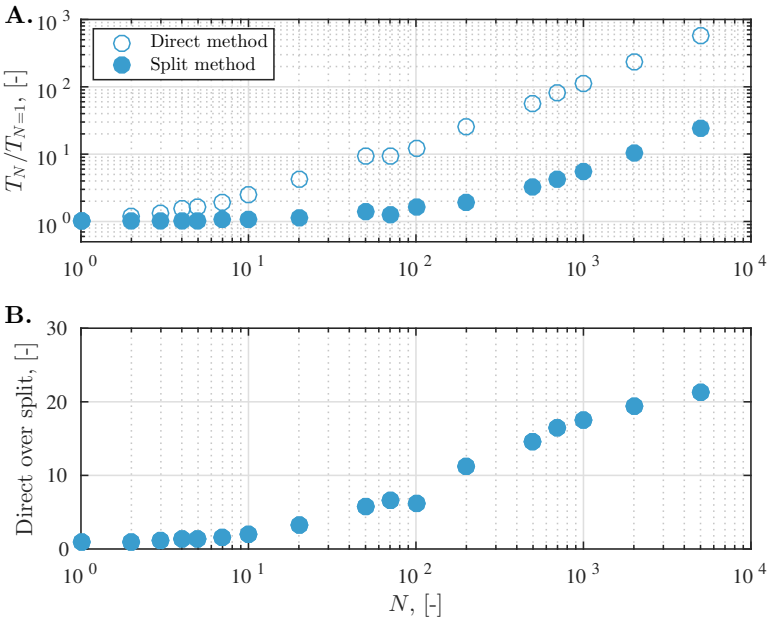

Figure B.6: A: The execution time of the irregular wave summation relative to the execution time for $N = 1$. B: The increase in execution time between the direct and the split methods.

The decrease in computational requirements between the two evaluation methods is a factor of 5 for $N = 100$ and the requirements decreased by more than an order of magnitude for $N = 1,000$[1]. Consequently, if 1,000 wave components are required for the generation of an irregular wave with equidistant discretisation of the spectrum, then a computational gain of 50 can be obtained by utilising a non-equidistant frequency discretisation in combination with the split method. Besides a faster execution of the numerical wave flume, this will also allow for the inclusion of second order effects in the incident wave field (Sharma and Dean, 1981; Madsen and Fuhrman, 2012), which would otherwise have been prohibitively expensive, because the number of second order components is N^2.

[1] It was observed that these ratios are hardware and/or compiler dependent.

APPENDIX C

Source Code History

This list is a reproduction of the information given at up to the revision covered by this document (see front page).

Table C.1: Summary of all code changes given per revision

Date	Revision	Description
2017-07-23 19:46:17	2113	Continue adding pressures to wave theories: Now chappelear and corrects for still water level
2017-07-15 22:07:26	r2112	Added solitary wave theory (Chappelear) and fixed bug in solitaryFirst
2017-07-14 18:01:19	r2111	Adding pressure distribution to stokesSecond
2017-05-17 17:12:03	r2107	Added a small interface correction to waveTheory.H
2017-05-07 21:53:13	r2106	Added utility to compute the (analytical) wave field in a multiple of points: sampleIncidentWaveField
2017-04-09 20:23:26	r2105	Added breeder version of catenary mooring line (v1606). Also coded pressure to StokesFirst
2017-01-15 18:50:23	r2104	Support for foam-extend-4.0
2016-11-08 18:41:24	r2103	waves2Foam available for v1606+
2016-10-13 16:00:22	r2102	Preparing for better treatment of KC; no functional change
2016-09-27 07:31:45	r2101	porousWaveFoam for OpenFoam-v3.0+ (only explicit advection of alpha.water)
2016-08-12 15:48:56	r2100	Added compatibility with OpenFoam-v3.0+ and solvers for foam-extend-3.2
2016-07-08 10:40:35	r2096	Minor change to pEqn.H for foam-extend-3.1 to support on-going developments. No functional change.
2016-07-01 22:41:30	r2095	Added support for OpenFoam-4.0.0. porousWaveFoam not created and sampling library broken in OpenFoam-4.0.0
2016-05-19 22:05:52	r2094	Support for stream function wave theory
2016-05-19 10:27:57	r2093	Debug overtopping: results of rhoU2A were bugged, not the overtopping rate itself.
2016-04-29 11:11:26	r2092	Minor cosmetic changes. Only functional change: Extruded faceSetToSTL always have outward pointing normal.
2016-04-04 14:52:42	r2091	Changed source location for OceanWave3D. New version resolves bugs with respect to wave generation with the use of a paddle signal. Otherwise no functional effects are expected.
2016-04-04 09:48:18	r2090	Resolved need for dual compilation, if WAVES_LIBBIN does not exist at the beginning of the compilation. WAVES_LIBBIN and WAVES_APPBIN are made manually.
2016-03-08 11:05:46	r2089	Minor downstream dependencies. No functional change
2016-02-03 10:58:03	r2088	Minor bug corrected in compilation procedure; effect on foam-extend branch
2016-02-02 10:50:34	r2087	Corrected minor bug in oceanWave3D-coupling.
2016-01-08 17:42:13	r2086	Update tutorial porousDamBreak + porousWaveFoam for OF2.3.0
2015-12-18 18:29:10	r2085	Added time control checks to oceanWave3D.C
2015-11-23 07:27:38	r2084	Added solver for OpenFoam-3.0.0
2015-11-20 18:50:30	r2083	Preparation of compatibility with OpenFoam-3.0.0 - do not use this revision
2015-10-15 22:09:38	r2082	Debugging of ThirdParty/Allwmake compile script
2015-10-15 21:17:44	r2081	Additional of OceanWave3D coupling + related tutorial + documentation in 'doc'
2015-10-01 20:00:40	r2080	src+utilities compiles with foam-extend-3.2. Solvers to follow
2015-09-16 09:27:48	r2079	Minor issues with options and file for solvers
2015-09-09 16:07:37	r2078	Added the possibility of using the sea level as vertical reference. Tutorials and setWaveParameters modified accordingly.
2015-08-24 17:10:42	r2077	Added '- phi' to the argument in stokesFifth.C
2015-07-27 09:44:25	r2073	Minor changes related to external wave forcing (e.g. nicer closing in solver + works with setWaveField)
2015-07-12 13:48:09	r2072	waves2Foam compiles on OF2.4 + waveFoam solver

2015-06-06 18:22:46	r2068	Added zero-crossing functionality + porosity model can be used for non-wave related things
2015-04-14 21:24:03	r2066	Streamlined the code (removed duplication) for the setWaveProperties and related classes. Change in input format in waveProperties.input.
2015-03-25 09:05:40	r2064	surfaceElevation now gives correct results on moving meshes in runTime (tested on foam-extend-3.1). Increase computational time.
2015-03-16 14:38:06	r2063	Small change to porosityCoefficients. NO functional change. Also, corrected bug in surfaceElevation.C, which affected foam-extend branch.
2015-03-12 15:14:46	r2062	Small change in the interface for the porosity module. No functional change.
2015-02-05 13:28:26	r2060	Added porousWaveFoam for foam-extend-3.1
2014-10-30 11:01:34	r2057	Resolving naming match between waves2Foam and 2+ OF-versions [DO NOT USE REVISION 2056]
2014-10-30 10:18:49	r2056	Made the porosity runTime-selectable. Currently only one method available. Commit includes derived modifications.
2014-10-24 22:03:27	r2055	Directional check on wave directions in wave theories. 3Dwaves tutorial corrected accordingly
2014-09-09 14:41:16	r2051	Small interface change in the external wave classes (constructs now with fvMesh as well)
2014-09-04 14:43:15	r2050	Modified options files so the compilation works with local installations of GSL
2014-09-04 11:36:01	r2049	Changes to the output location of some applications and libraries. Now uses WAVES_APPBIN/WAVES_LIBBIN
2014-06-21 18:04:47	r2045	Compiling on foam-extend-3.1, waveFoam 3.1 and waveFoam/-porousWaveFoam added directly from OF2.2.1 to OF2.2.2
2014-05-31 22:02:48	r2044	waveFoam support to foam-extend 3.0 + minor bug-fixes + new relaxation zone shape
2014-05-17 11:28:34	r2043	Minor maintanence, i.e. waveProperties.input in waveFlume tutorial.
2014-04-12 17:35:52	r2042	Corrected OF2.3.0 issue with the test case bejiBattjes
2014-04-06 12:33:09	r2041	Release to facilitate planned future open-source contribution(s) [No functional change in execution].
2014-03-27 21:17:38	r2040	Additional corrections to the tutorials to run in 2.3.0. Also corrected periodicSolitary, so the tutorial runs for recent cyclic definition.
2014-03-26 19:22:46	r2039	Corrected tutorials, such that they can actually execute (adding the 0.org-directory)
2014-03-20 11:11:40	r2038	Added a single post-processing functionality (writeIndexLocation)
2014-03-11 20:57:55	r2037	Updated structure of tutorials to easier accommondate future changes. Now runs for OF2.3 with alpha.water.
2014-03-01 19:05:46	r2036	Adjusted to be used with OF2.3. Furthermore, the porousDamBreak case now offers comparison with experiments. The tutorials are not yet adjusted to work with the changes from OF2.2 -> OF2.3.
2014-01-26 16:13:27	r2026	Compatibility of waves2Foam libraries and utilities with foam-extend-3.0. Still no solver (waveFoam)
2013-12-28 11:43:19	r2025	Porosity module with solvers (OF1.6, 2.1.0, 2.1.1, 2.2.0, 2.2.1) and a single tutorial.
2013-11-25 19:35:06	r2024	Modified the fatal error message in the compilation, such that it is easy to understand how to resolve the issue with a wrong WAVES_DIR environmental variable.
2013-10-24 09:11:15	r2022	Added a new relaxationScheme and a new relaxationShape
2013-10-07 20:36:24	r2021	Modified src/waves2Foam*/Make/options.* to include XVERSION flag
2013-09-15 09:37:28	r2020	Added phase focusing as an option for irregular waves.
2013-09-05 13:31:41	r2019	Trivial bug-fix, not affecting results (Thanks: Bjarne Jensen)
2013-09-03 10:38:12	r2018	Increased flexibility for setting wave spectra for JONSWAP.
2013-08-12 14:40:19	r2017	Minor bug-fix in src/waves2Foam/waveTheories/irregular/irregular/-irregular.C
2013-08-07 23:34:24	r2016	Faster evaluation of irregular waves
2013-08-06 21:59:56	r2015	A bit of code maintenance/simplification
2013-07-28 12:15:35	r2014	Fourth revision on code style change according to: www.openfoam.org/contrib/code-style.php. [No functionality change]
2013-07-27 22:03:15	r2013	Third revision on code style change according to: www.openfoam.org/contrib/code-style.php. [No functionality change]
2013-07-27 19:31:35	r2012	Second revision on code style change according to: www.openfoam.org/contrib/code-style.php. [No functionality change]
2013-07-22 22:53:39	r2011	First revision on code style change according to: www.openfoam.org/contrib/code-style.php. [No functionality change]
2013-07-15 13:09:08	r2010	Corrected missing files in src/waves2FoamProcessing/Make/files
2013-07-13 11:32:48	r2009	waves2Foam now compiles under OF2.2.1
2013-07-09 15:12:25	r2008	Added an additional point distribution (user-defined) for wave gauges
2013-07-07 19:28:45	r2007	Release of pre- and post-processing utilities, overtopping sampling and modified tutorials.
2013-07-07 18:37:43	r2004-r2006	Temporary revision: Restructuring the src-directory
2013-07-07 17:41:18	r2003	Bug in bichromatic second.
2013-06-11 20:46:14	r2000	Resolve incompatibility between version 2.2.0 and 2.2.x for waveFoam header files
2013-06-09 09:13:39	r1997	Added a small script for the creation of new wave theories in waves2Foam/doc/templateWaveTheory
2013-06-08 21:11:23	r1996	Updated compilation system
2013-06-07 21:38:02	r1991-r1995	Temporary commit during updating to new SVN
2012-12-03 12:52:40	r1984	Added second order bichromatic waves. Set numerical beach type to "Empty" as defalt.
2012-11-29 15:10:18	r1983	Added wave theory. Automatic enhanced write precision for cnoidal waves parameters. Slight change to waveTheory interface.

2012-11-23 15:52:01	r1982	Changed setWaveParameters so (i) parameters are read from waveProperties.input and written to waveProperties and (ii) it does not depend on the mesh. The latter speeds up the execution (a lot!).
2012-11-18 21:48:17	r1981	Added a check for validity of second order stokes in the properties file
2012-10-31 15:29:27	r1980	Bug in compiling setWaveParameters for OFVERSION >= 17. Added pre-processor statement
2012-10-31 14:38:18	r1979	(i) Bug in bichromaticFirstProperties.C. (ii) Cosmethics in the *Properties files, when writing waveProperties-file
2012-10-18 17:00:21	r1977	(i) Added solvers for 2.1. (ii) Changed the approach used by setWaveParameters to write waveProperties file. (iii) Updated headers with publication data on the journal article. (iv) Added the file sourceCodeStructure_r1923.svg in the doc directory.
2012-10-17 12:04:01	r1976	Removed all pre-processing if's related to version 15. Altered the OFVERSION pre-processor to work on e.g. 211 and 160 rather than 21 and 16.
2012-10-01 11:49:20	r1975	Modified the tutorials, so they are also running under OF2.1
2012-09-10 21:29:21	r1973	Corrected infinite loop in the computation of the wave number, when kh > 60 (extreme deep water).
2012-08-16 09:44:50	r1972	Added an option for a local sealevel in potentialCurrent
2012-07-13 20:51:52	r1969	Added the following: (i) runTime selection of relaxation zone weights, (ii) a local correction to relaxation weight based on the local Courant number, (iii) update of relaxationZoneLayout to show the weight and (iv) added interface for numerical beach in UEqn.H but still not functional.
2012-06-08 12:09:21	r1967	Added some post-processing utilities to be used in matlab and modified the misc/matlab file structure. Minor bug-fix in generateStreamFile.m so the output file is consistent with the format in waves2Foam.
2012-05-16 10:44:43	r1966	Corrected bug in matlab script for stream function theory (previous results correct, if the program could finalise!)
2012-05-15 12:01:18	r1965	Additional output time control for the surfaceElevation utility. Especially needed under functionObject functionality
2012-05-10 14:37:47	r1961	Added a optional starting time for the surface elevation sampling
2012-05-09 14:18:23	r1960	Added phase-lag (phi_) in streamFunction.C
2012-05-03 13:22:43	r1953	Added functionObject functionality for sampledSurfaceElevation. Included in the tutorial waveFlume.
2012-05-02 10:11:01	r1952	Modification to Allwmake and src/Allwmake so "sed" also works under Mac OSX.
2012-04-30 21:28:20	r1951	Added a post-processing utility, which can be used to extract the surface elevation. As of now, version 1.5 is no longer supported.
2012-04-11 10:13:10	r1947	Bug in Allwmake script
2012-03-26 21:45:20	r1945	Changed the way setWaveProperties output irregular wave properties. Adjusted irregular waveTheory accordingly.
2012-03-17 15:56:02	r1944	Solved bug in convexPolyhedral. Furthermore added an addition relaxation shape (semiCircular)
2012-03-13 16:13:42	r1943	Minor changes
2012-02-19 14:45:14	r1940	Added the possibility of a wind vector, which is constant in space and time.
2012-01-29 19:15:41	r1938	Changed the Make/files and Make/options to have the output files in FOAM_USER_LIBBIN and FOAM_USER_APPBIN.
2012-01-28 15:55:34	r1937	Added a Pierson-Moskowitz spectrum
2012-01-27 13:28:00	r1936	Added framework for irregular wave spectra - properties and wave theory
2012-01-24 13:20:58	r1935	Small changes to src/Make/files and the overall Allwmake script in order to avoid the recompilation of waves2Foam on non-dev/ext versions of OF
2012-01-19 16:50:48	r1934	Modified the code, so PI can be used both in pre- and post-2.0 versions of OpenFoam. It is replaced by PI_(4.0*atan(1.0)) a few neccesary places as a protected member variable.
2011-12-04 20:07:35	r1932	Added a tutorial showing how to generate a standing wave from a fully reflecting sea wall
2011-11-22 10:02:44	r1929	Added a tutorial utilising the cylindrical relaxation zone
2011-11-19 01:01:19	r1928	Added "waveFoam" for 1.7, modified Allwmake script and added necessary files to tutorials due to above change.
2011-11-16 19:22:22	r1937	setWaveParameters now also works for <tt>cnoidalFirst</tt>
2011-11-14 20:04:08	r1926	Added matlab tools for cnoidal and stream function waves
2011-11-14 17:52:52	r1925	Added a 3D wave tank tutorial
2011-11-10 11:30:30	r1923	Initial release of waves2Foam - all files