

Users Guide for multi-phase DualSPHysics code (gas-liquid)



DualSPHysics v4.0

March 2017

dualsphysics@gmail.com

Athanasios Mocos

Benedict D. Rogers

Peter K. Stansby

Abstract

This guide documents the multi-phase gas-liquid module for the Smoothed Particle Hydrodynamics (SPH) code DualSPHysics. This manuscript describes the additional functions inserted in the code to allow for the simulation of multiple phases. The code has also been extended to work with bubbly flows and new pre-processing options have been added to facilitate the end user.

Acknowledgements

The authors gratefully acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) [EP/H003045/1] and the Natural Environment Research Council (NERC) [NE/J009202/1] for funding this work. The authors would also like to thank colleagues at the University of Manchester and at the Universidad de Vigo and helping in the development of this work.

Contents

Abstract	2
Acknowledgements	2
Contents	3
Introduction	4
Code Changes	5
Command Line.....	6
Xml Input File: Additional Parameters for gas-liquid	7
Output	8
Test Cases	9
Appendix.....	12
References	16

Introduction

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian meshless method with a wide range of applications. SPH is widely used for Computer Fluid Dynamics (CFD), with the particles representing the fluids and structures. SPH has the ability to efficiently model interfaces, both for complex, rapidly-changing flows and for longer slower applications such as waves. This makes it ideal for use with multi-phase applications such as wave impact on offshore structures and potentially explosive slug flow in the nuclear industry. However, obtaining sufficient accuracy requires a large number of particles which is computationally expensive.

The runtime can be decreased by using hardware acceleration and parallel computing. Graphics Processing Units (GPUs) provide a massively parallel architecture with low cost making them suitable for SPH simulations. DualSPHysics (Crespo et al., 2015) is an SPH code written in C++ and CUDA that takes advantage of the GPUs and can also efficiently carry out simulations on a CPU using OpenMP.

The DualSPHysics code is an open source project mainly led by the Environmental Physics Laboratory (EPHYSLAB) from Universidade de Vigo (Spain) and the School of Mechanical, Aerospace and Civil Engineering (MACE) from The University of Manchester (UK). The DualSPHysics code originates from SPHysics (www.sphysics.org), which is an open-source SPH model developed by researchers at the Johns Hopkins University (US), the University of Vigo (Spain), the University of Manchester (UK) and the University of Rome, La Sapienza.

The DualSPHysics code, in its original release, can only deal with single-phase flows. This limits the range of applications as there are many flows where the presence of a second phase is essential or a multitude of fluids are mixing. In addition, it is more suited to violent or gravity flows and modelling liquids.

This release of DualSPHysics adds support for multiple fluids within the code via a C++ structure that allows the code to load different attributes for each phase (Mokos et al., 2015). These fluids can be either liquids or gases as a multi-phase model has been incorporated including a multi-phase version of the shifting algorithm (Mokos et al., 2016). A surface tension model (Hu and Adams, 2006) has also been incorporated allowing the code to now deal with bubbly flows.

For the end user, the new changes and their respective parameters can be defined via the input file of the GenCase pre-processing tool without requiring C++ knowledge to alter the code. The changes have been incorporated in both the CPU and the GPU paths while still allowing the user to model a single-phase test case. The changes are optimised but an impact to performance is unavoidable due to the increased complexity of the code (Mokos et al. 2015).

This document details the implementation of the multi-phase model for DualSPHysics (<http://www.dual.sphysics.org/>) and is a supplement to the main DualSPHysics v4.0 guide (<http://www.dual.sphysics.org/index.php/downloads/>) which contains the necessary information on the pre- and post-processing tools as well as compiling and running the code.

To reference the multi-phase DualSPHysics code, please reference the peer-reviewed journal papers (Mokos et al., 2015) and (Mokos et al., 2016).

Code Changes

This section of the document details the main functions and files that have been altered or created to incorporate the multi-phase changes. Not all functions are described here, but a complete list of the functions can be found through the Doxygen documentation of DualSPHysics. New functions have the suffix MP at the end of their name while changes within the code are marked with $\mathbb{E}\mathbb{E}/\text{MP}$.

As a basic structure, the code has been divided into two branches, the original single-phase and the new multi-phase branch allowing for greater flexibility. The philosophy of the multi-phase code is described in (Mokos et al., 2015). Note that the multi-phase branch can still simulate single-phase flows but it will be slightly slower. It is also important to clarify that the shifting model used in the original code ((Lind et al., 2016) with a different free surface correction) is different to the one in the multi-phase code (Mokos et al., 2016).

JCfgRun.cpp	Loads command line arguments for the multi-phase flow	
JSph.cpp	LoadCaseConfig	Loads multi-phase attributes from the input file (LoadCaseConfig), updates density and mass for different phases (UpdateRhopMP and UpdateMassMP) and prints multi-phase data (VisuMP)
	UpdateRhopMP	
	UpdateMassMP	
	VisuMP	
JSphCpuSingle.cpp JSphGpuSingle.cpp	ComputeStep_Sym	Switches between the two branches (ComputeStep_Sym or_Verlet) and starts the computation of the particle interaction (Interaction_ForcesMP) and the shifting model (Run_ShiftingMP)
	ComputeStep_Ver	
	Interaction_ForcesMP	
	RunShifting_MP	
JSphCpu.cpp	Interaction_ForcesMP	Computes particle interactions (Interaction_ForcesMP), the shifting model (Run_ShiftingMP) and updates the particles' data (Compute_VerletMP and Compute_SymplecticPre(Corr)MP)
	PreInteraction_ForcesMP	
	Compute_VerletMP	
	Compute_SymplecticPreMP	
	Compute_SymplecticCorrMP	
	Run_ShiftingMP	
JSphGpu.cpp	ConstantDataUp	Uploads multi-phase data (Constant(Particles)DataUp) and begins GPU computation of preliminary interactions (PreInteraction_ForcesMP) and time stepping scheme (ComputeVerletMP, ComputeSymplecticPre(Corr)MP)
	ParticlesDataUp	
	PreInteraction_ForcesMP	
	ComputeVerletMP	
	ComputeSymplecticPreMP	
	ComputeSymplecticCorrMP	
JSphGpu_ker.cu	KerInteractionForcesFluidMP	Contains the kernels for particle interaction, the shifting model and the time stepping scheme
	KerInteractionForcesBoundMP	
	KerRunShiftingMP	
	KerComputeVerletMP	
	KerComputeStepSymplecticPreMP	
Types.h	Contains the data structures for the multi-phase data	

Table 1: Main files and functions changed for the multi-phase flow

Command Line

The following commands can be used to control aspects of the multi-phase DualSPHysics through the command line. The same options are also present in the GenCase input file but using the command lines will override the file.

<i>-multiphase:<float></i>	Sets the execution to the multi-phase branch. Specify cohesion coefficient for the Colagrossi and Landrini multi-phase model.
<i>-singlephase</i>	Sets the execution to the single-phase branch
<i>-shifting:<int></i>	Added value to specify shifting coefficient
<i>-shiftgradx:<float></i> <i>-shiftgrady:<float></i> <i>-shiftgradz:<float></i>	Expected particle concentration gradient at the free surface in the defined direction. Only used in the liquid phases. Should be applied only to longer, non-violent flows.
<i>-incx:<float></i> <i>-incy:<float></i>	Increase the domain size in the x- or y-direction. Same as the IncZ option in the original DualSPHysics.

Table 2: Additional commands introduced for the multi-phase model

An example of running a case file on GPU with the symplectic time step algorithm and the Wendland kernel is provided. The multi-phase includes a cohesion coefficient of 0.2 and a shifting coefficient of -6.:

- Single-phase: DualSPHysics_v4.0 -gpu -singlephase path/casefile -dirout path/output -symplectic -wendland
- Multi-phase: DualSPHysics_v4.0 -gpu -multiphase:0.2 path/casefile -dirout path/output -symplectic -wendland -shifting:-6

Xml Input File: Additional Parameters for gas-liquid

Both the gas-liquid multi-phase and the single-phase branch use the same GenCase program to produce an input file. However, the multi-phase code requires some additional data, while some options are deprecated. This section will highlight these changes for each different section of the GenCase xml file. If a section is not mentioned no changes are needed.

➤ *<constantsdef>*

The *<speedsystem>*, *<coefssound>*, *<speedsound>* options are not used for the multi-phase code. The speed of sound of the fluids is set through the *<properties>* section instead.

➤ *<properties>*

A *<properties>* section is required by the multi-phase code. Each different fluid must be represented by a separate *<property>* an example structure of which is presented here:

```
<property name="water">           (Specify fluid name)
  <rho0 value="1000"/>           (Initial density of the fluid)
  <gamma value="7"/>             (Isentropic factor)
  <cs0 value="62.485"/>          (Speed of sound value)
  <visco value="0.1"/>           (Viscosity coefficient: laminar or artificial)
  <phasetype value="0"/>         (Fluid type: 0 for liquids, 1 for gases)
  <surfcoef value="0"/>          (Surface Tension coefficient: optional)
```

</property>

To identify a specific volume as a fluid a *<link>* must be created between the *<property>* and the mark of the volume (*mk*). This must be done for every *mk* of the simulation, including boundaries. Multiple marks can have the same property. An example is presented below:

<links>

```
<link mkbound="0,10" property="water"/>
<link mkfluid="0" property="water" />
<link mkfluid="1" property="air" />
```

</links>

When linking properties to the boundaries it is recommended that the heavier fluid of the simulation is used.

➤ *<parameters>*

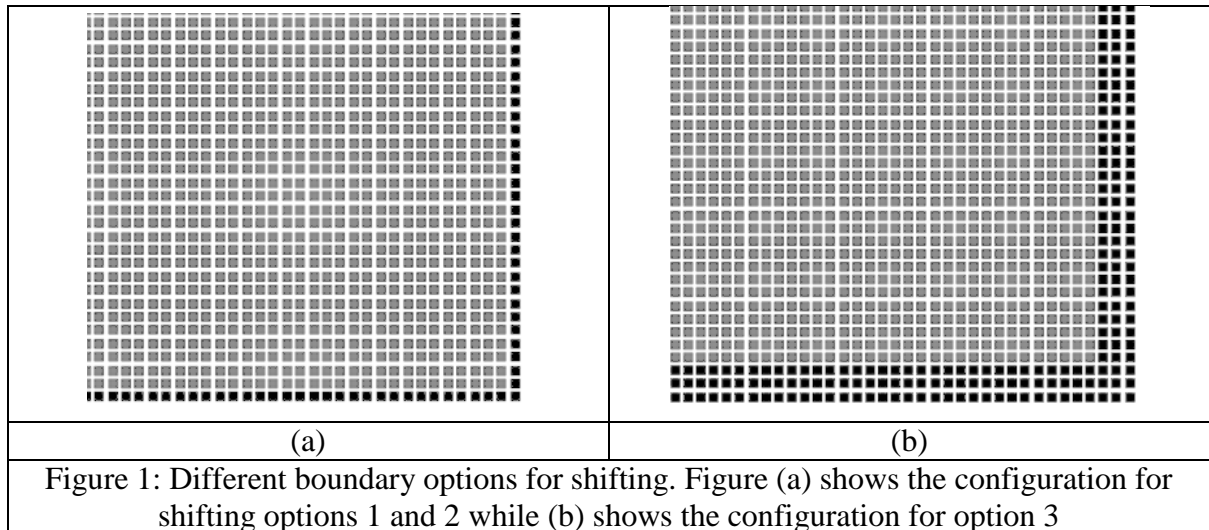
1. A new parameter *<FlowType>* is introduced to switch between using the single- and multi-phase branches.

```
<parameter key="FlowType" value="2" comment="Select Single-phase (1) or multi-phase flow(2)" />
```

2. The parameter *<Visco>* is deprecated. Viscosity is altered through the *<properties>* section as above.
3. The parameter *<Shifting>* controls the new shifting model. Option 3 is the most accurate, but requires multiple boundary layers (2-3) to avoid particles exiting the system. Use option 1 or 2 if only one boundary layer is used. An example is shown in Figure 1. When modelling gases it is recommended that some form of shifting is active.
4. The recommended values for the parameter *<ShiftTFS>* are 1.4-1.6 for 2-D flows and 2.5-2.7 for 3-D flows.

5. New parameters `<ShiftNormalGradX>`, `<ShiftNormalGradY>`, `<ShiftNormalGradZ>` are introduced. Used for slow, non-violent flows, they are a prediction of the expected normal shifting gradient in the free surface; more details in (Mokos et al., 2016).
`<parameter key="ShiftNormalGradX" value="0" comment="Expected concentration gradient on the normal to the liquid free surface. Only used for slow, non-violent flows" />`
6. The new parameter `<MPCoefficient>` is introduced. This parameter is the cohesion coefficient from the multi-phase model used here (Mokos et al., 2016) (Colagrossi and Landrini, 2003). Only used when modelling liquid-gas flows.
`<parameter key="MPCoefficient" value="0" comment="Multi-phase cohesion coefficient" />`
7. Two new parameters `<IncX>` and `<IncY>` are introduced. Their function is the same as existing parameter `<IncZ>` but on the x- and y-axis respectively.
`<parameter key="IncX" value="0.1" comment="Increase of X+" units_comment="decimal" />`

An example XML file detailing the creation of a dam break simulation with air and water can be found in the Appendix.



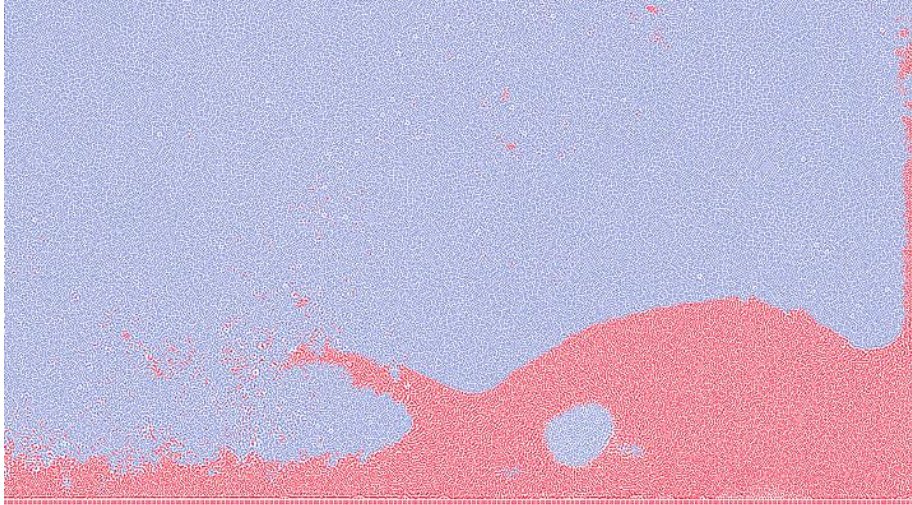
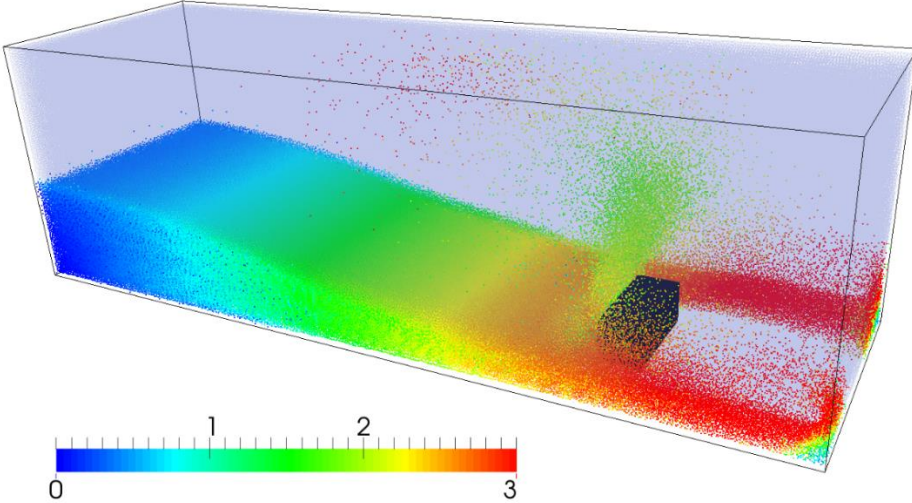

Output

The output has been expanded to include the correct pressure for multiple phases. Outputting a .csv or .vk file directly from DualSPHysics with the commands `–savecsv` and `–savevtk` will automatically include a pressure array. If binary files are selected as output (the default option) then the command `–vars:press` should be used with the PartVTK post-processing tool to show the pressure array.

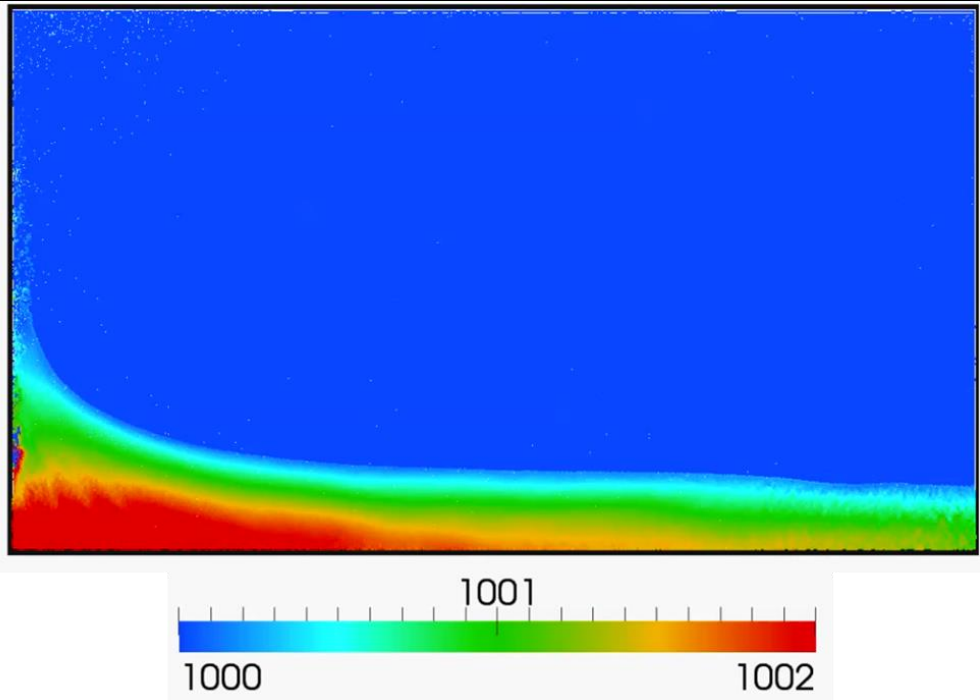
Example:

Test Case Snapshots

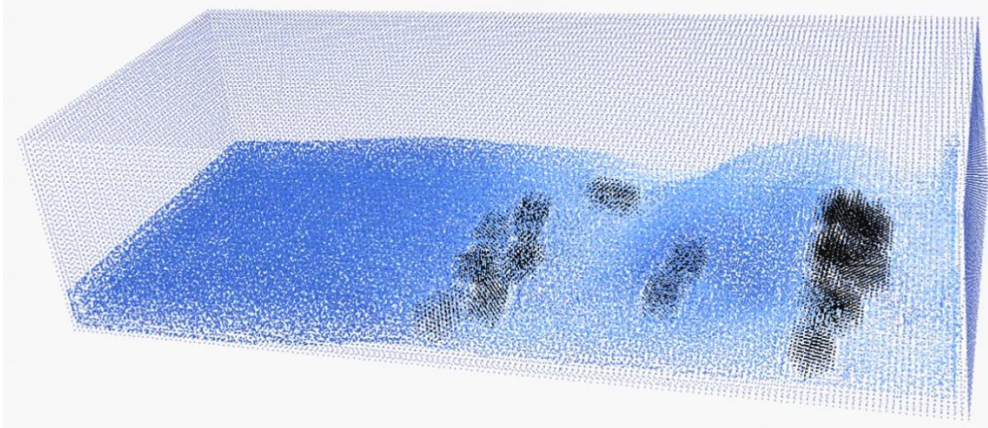
The following test cases are included in the multi-phase DualSPHysics code

<p>Case 1: Dam Break (2s)</p>	
<p>Case 2: Obstacle Impact (0.5s)</p>	
<p>Case 3: Wet Dam Break (0.343s)</p>	

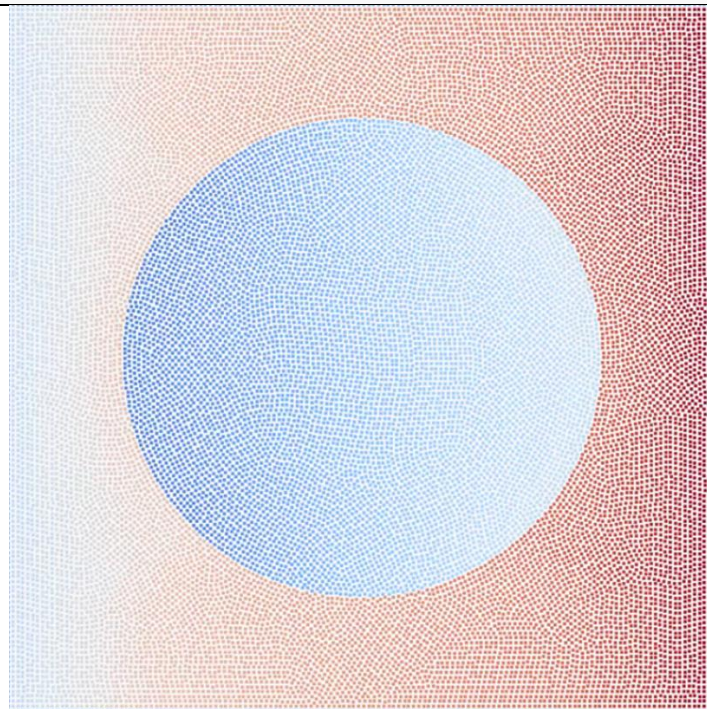
Case 4:
Sloshing
Tank
(2.4s)



Case 5:
DEM
Interaction
(4.2s)



Case 6:
Surface
Tension
(0.5s)



Appendix – Example XML for 2-D Multi-Phase Dam Break

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<case date="09/07/2013 16:42:25">
  <casedef>
    <constantsdef>
      <lattice bound="1" fluid="1"/>
      <gravity x="0" y="0" z="-9.81" comment="Gravitational acceleration" units_comment="m/s^2" />
      <rhof0 value="1000" comment="Reference density of the fluid" units_comment="kg/m^3" />
      <hswl value="0" auto="true" comment="Maximum still water level to calculate speedofsound using coefsound"
units_comment="metres (m)" />
      <gamma value="7" comment="Polytropic constant for water used in the state equation" />
      <speedsystem value="0" auto="true" comment="Maximum system speed (by default the dam-break propagation is used)" />
      <coefsound value="2" comment="Coefficient to multiply speedsystem" />
      <speedsound value="0" auto="true" comment="Speed of sound to use in the simulation (by default
speedofsound=coefsound*speedsystem)" />
      <coefh value="1.0" comment="Coefficient to calculate the smoothing length (h=coefh*sqrt(3*dp^2) in 3D)" />
      <cflnumber value="0.1" comment="Coefficient to multiply dt" />
    </constantsdef>
    <mkconfig boundcount="20" fluidcount="2"/>
    <geometry>
      <definition dp="0.02">
        <pointmin x="-5.5" y="0" z="-2.5" />
        <pointmax x="5.5" y="0" z="2.5" />
      </definition>
      <commands>
        <mainlist>
          <setshapemode>dp | bound</setshapemode>
          <setdrawmode mode="full" />
          <setmkfluid mk="0" />
        </mainlist>
        <drawbox>
          <boxfill>bottom | left | right | front | back</boxfill>
          <point x="-2" y="0" z="-2" />
          <size x="1" y="0" z="2" />
        </drawbox>
        <setmkfluid mk="1" />
      </drawbox>
      <drawbox>
        <boxfill>bottom | left | right | front | back</boxfill>
        <point x="-2" y="0" z="0" />
        <size x="4" y="0" z="2" />
      </drawbox>
    </geometry>
  </casedef>
</case>
```

```

        </drawbox>
    <drawbox>
        <boxfill>bottom | left | right | front | back</boxfill>
        <point x="-1" y="0" z="-2" />
        <size x="3" y="0" z="2" />
    </drawbox>
    <setdrawmode mode="wire" />
    <setmkbound mk="0" />
    <drawbox>
        <boxfill>top | bottom | left | right | front | back</boxfill>
        <point x="-2.0" y="0" z="-2.0" />
        <size x="4.0" y="0" z="4" />
    </drawbox>
    <drawbox>
        <boxfill>top | bottom | left | right | front | back</boxfill>
        <point x="-2.02" y="0" z="-2.02" />
        <size x="4.04" y="0" z="4.04" />
    </drawbox>
    <drawbox>
        <boxfill>top | bottom | left | right | front | back</boxfill>
        <point x="-2.04" y="0" z="-2.04" />
        <size x="4.08" y="0" z="4.08" />
    </drawbox>
</mainlist>
</commands>
</geometry>
<properties>
    <links>
        <link mkbound="0" property="water"/>
        <link mkfluid="0" property="water" />
        <link mkfluid="1" property="air" />
    </links>
    <property name="water">
        <rho0 value="1000"/>
        <gamma value="7"/>
        <cs0 value="50"/>
        <visco value="0.001"/>
        <phasetype value="0"/>
        <surfcoef value="0"/>
    </property>
    <property name="air">

```

```

        <rhop0 value="1.18"/>
        <gamma value="1.4"/>
        <cs0 value="400"/>
        <visco value="0.0001"/>
        <phasetype value="1"/>
        <surfcoef value="0"/>
    </property>
    <property name="heavy_water">
        <rhop0 value="1020"/>
        <gamma value="7"/>
        <cs0 value="62.485"/>
        <visco value="0.0001"/>
        <phasetype value="0"/>
        <surfcoef value="0"/>
    </property>
</properties>
</casedef>
<execution>
    <parameters>
        <parameter key="PosDouble" value="1" comment="Precision in particle interaction 0:Simple, 1:Double, 2:Uses and saves
double (default=0)" />
        <parameter key="FlowType" value="2" comment="Select Single-phase (1) or multi-phase flow(2)" />

        <parameter key="StepAlgorithm" value="1" comment="Step Algorithm 1:Verlet, 2:Symplectic (default=1)" />
        <parameter key="VerletSteps" value="40" comment="Verlet only: Number of steps to apply Euler timestepping
(default=40)" />
        <parameter key="Kernel" value="2" comment="Interaction Kernel 1:Cubic Spline, 2:Wendland (default=2)" />

        <parameter key="ViscoTreatment" value="1" comment="Viscosity formulation 1:Artificial, 2:Laminar+SPS (default=1)" />
        <parameter key="Visco" value="0.0" comment="Viscosity value" /> % Note alpha can depend on the resolution. A value of
0.01 is recommended for near irrotational flows.
        <parameter key="ViscoBoundFactor" value="1" comment="Multiply viscosity value with boundary (default=1)" />
        <parameter key="DeltaSPH" value="0.1" comment="DeltaSPH value, 0.1 is the typical value, with 0 disabled (default=0)"
/>

        <parameter key="Shifting" value="1" comment="Shifting mode 0:None, 1:Ignore bound, 2:Ignore fixed, 3:Full
(default=0)" />
        <parameter key="ShiftCoef" value="-6" comment="Coefficient for shifting computation (default=-2)" />
        <parameter key="ShiftTFS" value="1.5" comment="Threshold to detect free surface. Typically 1.5 for 2D and 2.75 for 3D
(default=0)" />
        <parameter key="MPCoefficient" value="0.1" comment="Colagrossi and Landrini multiphase coefficient." />

```

```

        <parameter key="RigidAlgorithm" value="1" comment="Rigid Algorithm 1:SPH, 2:DEM (default=1)" />
        <parameter key="FtPause" value="0.0" comment="Time to freeze the floatings at simulation start (warmup) (default=0)"
units_comment="seconds" />
        <parameter key="CoefDtMin" value="0.05" comment="Coefficient to calculate minimum time step
dtmin=coefdtmin*h/speedsound (default=0.05)" />
        <parameter key="DtIni" value="0.0000001" comment="Initial time step (default=h/speedsound)" units_comment="seconds"
/>
        <parameter key="DtMin" value="0.00000001" comment="Minimum time step (default=coefdtmin*h/speedsound)"
units_comment="seconds" />
        <parameter key="#DtFixed" value="DtFixed.dat" comment="Dt values are loaded from file (default=disabled)" />
        <parameter key="DtAllParticles" value="0" comment="Velocity of particles used to calculate DT. 1:All, 0:Only
fluid/floating (default=0)" />
        <parameter key="TimeMax" value="0.1" comment="Time of simulation" units_comment="seconds" />
        <parameter key="TimeOut" value="0.01" comment="Time out data" units_comment="seconds" />
        <parameter key="IncZ" value="0" comment="Increase of Z+" units_comment="decimal" />
        <parameter key="PartsOutMax" value="1" comment="%/100 of fluid particles allowed to be excluded from domain
(default=1)" units_comment="decimal" />
        <parameter key="RhopOutMin" value="0" comment="Minimum rhop valid (default=700)" units_comment="kg/m^3" />
        <parameter key="RhopOutMax" value="1300" comment="Maximum rhop valid (default=1300)" units_comment="kg/m^3" />

    </parameters>
</execution>
</case>

```

References

- COLAGROSSI, A. & LANDRINI, M. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191, 448-475. DOI: 10.1016/S0021-9991(03)00324-3
- CRESPO, A. J. C., DOMINGUEZ, J. M., ROGERS, B. D., GOMEZ-GESTEIRA, M., LONGSHAW, S., CANELAS, R., VACONDIO, R., BARREIRO, A. & GARCIA-FEAL, O. 2015. DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications*, 187, 204-216. DOI 10.1016/j.cpc.2014.10.004
- HU, X. Y. & ADAMS, N. A. 2006. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213, 844-861. DOI: 10.1016/j.jcp.2005.09.001
- LIND, S. J., STANSBY, P. K. & ROGERS, B. D. 2016. Incompressible–compressible flows with a transient discontinuous interface using smoothed particle hydrodynamics (SPH). *Journal of Computational Physics*, 309, 129-147. DOI: 10.1016/j.jcp.2015.12.005
- MOKOS, A., ROGERS, B. D., STANSBY, P. & DOMINGUEZ, J. M. 2015. Multi-phase SPH Modelling of Violent Hydrodynamics on GPUs. *Computer Physics Communications*. DOI: 10.1016/j.cpc.2015.06.020
- MOKOS, A., ROGERS, B. D. & STANSBY, P. K. 2016. A multi-phase particle shifting algorithm for SPH simulations of violent hydrodynamics with a large number of particles. *Journal of Hydraulic Research*, 1-20. DOI: 10.1080/00221686.2016.1212944