Multi-phase Modelling of Violent Hydrodynamics Using Smoothed Particle Hydrodynamics (SPH) on Graphics Processing Units (GPUs)

A thesis is submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2013

Athanasios Mokos

School of Mechanical, Aerospace and Civil Engineering

Contents

List of Figures

List of Tables

Abstract

Multi-phase Modelling of Violent Hydrodynamics using Smoothed Particle Hydrodynamics (SPH) on Graphics Processing Units (GPUs)

A thesis is submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

Athanasios Mokos, The University of Manchester, 2013

This thesis investigates violent air-water flows in two and three dimensions using a smoothed particle hydrodynamics (SPH) model accelerated using the parallel architecture of graphics processing units (GPUs). SPH is a meshless Lagrangian technique for CFD simulations, whose major advantage for multi-phase flows is that the highly nonlinear behaviour of the motion of the interface can be implicitly captured with a sharp interface. However, prior to this thesis performing multi-phase simulations of large scale air-water flows has been prohibitive due to the inherent high computational cost.

The open source code DualSPHysics, a hybrid central processing unit (CPU) and GPU code, is heavily modified in order to be able to handle flows with multiple fluids by implementing a weakly compressible multi-phase model that is simple to implement on GPUs. The computational runtime shows a clear improvement over a conventional serial code for both two- and three dimensional cases enabling simulations with millions of particles. An investigation into different GPU algorithms focuses on optimising the multi-phase SPH implementation for the first time, leading to speedups of up to two orders of magnitude compared to a CPU-only simulation. Detailed comparison of different GPU algorithms reveals a further 12% improvement on the computational runtime.

Enabling the modelling of cases with millions of fluid particles demonstrates some previously unreported problems regarding the simulation of the air phase. A new particle shifting

algorithm has been proposed for multi-phase flows enabling the air, initially simulated as a highly compressible liquid, to expand rapidly as a gas and prevent the formation of unphysical voids. The new shifting algorithm is validated using dam break flows over a dry bed where good agreement is obtained with experimental data and reference solutions published in the literature. An improvement over a corresponding single-phase SPH simulation is also shown.

Results for dam break flows over a wet bed are shown for different resolutions performing simulations that were unfeasible prior to the GPU multi-phase SPH code. Good agreement with the experimental results and a clear improvement over the single-phase model are obtained with the higher resolution showing closer agreement with the experimental results. Sloshing inside a rolling tank was also examined and was found to be heavily dependent on the viscosity model and the speed of sound of the phases. A sensitivity analysis was performed for a range of different values comparing the results to experimental data with the emphasis on the pressure impact on the wall.

Finally, a 3-D gravity-driven flow where water is impacting an obstacle was studied comparing results with published experimental data. The height of the water at different points in the domain and the pressure on the side of the obstacle are compared to a state-of-the-art single-phase GPU SPH simulation. The results obtained were generally in good agreement with the experiment with closer results obtained for higher resolutions and showing an improvement on the single-phase model.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

[1] http://www.campus.manchester.ac.uk/medialibrary/policies/intellectual-property.pdf
[2] http://www.manchester.ac.uk/library/aboutus/regulations

Acknowledgements

This thesis would not exist if not for the emotional and scientific support of the following people:

- I would like to express my appreciation to Dr. Benedict D. Rogers for his continuous support and supervision providing invaluable assistance both with mathematical issues and with matters such as presentations, writing academic papers and critical analysis of research problems.
- I am grateful to Professor Peter K. Stansby for his invaluable comments continuous support and guidance for the duration of my studies.
- I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for the financial support of this research and Dr. Benedict D. Rogers and Professor Peter K. Stansby for arranging this fund.
- I am thankful to all the members of the SPH and Hydrodynamics- expert group for their continuous support throughout weekly meetings and especially to Mr. Georgios Fourtakas for his continuous support for both numerical and theoretical matters and Dr. Stephen Longshaw for his assistance in computational issues.
- I would like to thank the University of Manchester academic, support and IT staff for providing excellent and informative workshops and for their technical and administrative support.
- I would like to thank the people at University of Vigo for their support with using the DualSPHysics code as well as their hospitality, with special acknowledgements for Mr. Jose M. Dominguez and Dr. Alejandro C. Crespo. I would also like to thank Dr. Steven Lind of Manchester Metropolitan University for his assistance.
- I will not forget the people I was sharing an office with at the university and I thank them for their daily support and for brightening up my day.
- My thanks go to all the friends I have made during my stay in Manchester and especially to Mr. Ioannis Asproulias, Miss Joanna Peppers and Mr. Antonios Xenakis for their invaluable help and support.
- And last, but not least, I would like to express my gratitude to my family, who provided me with enormous emotional support and encouragement over the years.

Nomenclature and Glossary

Symbol	Definition
A	Arbitrary function
A_s	Shifting parameter
c_s	Speed of sound
C	Particle concentration
C_{CFL}	CFL number
C_{sh}	Empirical shifting constant
dx	Particle spacing
d_0	Height of the water layer
D	Diffusion coefficient
e	Energy
f	Applied forces
$\mathbf{g}$	Gravity force
h	Smoothing length
h_0	Initial height of the water column
H	Height of the domain
i	Particle where the interpolation is performed
j	Neighbouring particle
$\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$	Unit vectors
L	Length of the domain
m	Particle mass
M	Mach number
$\mathbf{n}$, $\mathbf{s}$, $\mathbf{b}$	Normal, tangent and binormal vectors to the free surface
N	Number of neighbouring particles
O	Truncation error
p	Particle pressure
q	$\mathbf{r}_{ij}/h$
$\mathbf{r}$	Coordinate vector
$\mathbf{R}$	Shifting vector
Re	Reynolds number
t	Time
$\mathbf{u}$	Velocity
W	Smoothing kernel
$\tilde{W}$	Zeroth order corrected kernel
X	Background pressure
a	Artificial viscosity coefficient
a_s	Shifting magnitude
$\bar{a}$	Cohesion coefficient
a_D	Kernel normalisation factor
β	Second artificial viscosity parameter
γ	Polytropic index
δ_ε	Dirac function
$\delta\mathbf{r}_s$	Shifting distance
Δt	Time step

ε	Effective region of the Dirac function
μ	Dynamic viscosity
ν_0	Laminar kinematic viscosity
Π	Artificial viscosity term
ρ	Particle density
ρ_0	Reference density
Ω	Domain of interest in the interpolation of function A

Term	Definition
Artificial Viscosity	Additional term in the momentum equation simulating the viscous forces among the particles
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture is a framework for programming GPUs
DualSPHysics	Hybrid CPU-GPU code for fluid simulations using SPH
ISPH	Incompressible Smoothed Particle Hydrodynamics
GPU	Graphics Processing Units are dedicated graphic units used here for hardware acceleration
MPI	Message Parsing System transferring data between CPUs
OpenCL	Open Computing Language is a framework for programming GPUs
OpenMP	Open Multi-Processing is a protocol allowing shared memory programming between CPUs
SM	Streaming Multiprocessors are a basic computing unit of the GPU
Smoothing Kernel	Interpolating function used in SPH
Smoothing Length	Defines size of the kernel influence
SPH	Smoothed Particle Hydrodynamics
SPHERIC	SPH European Research Interest Community
Tensile Instability	SPH phenomenon where negative pressures cause particle attractions
WCSPH	Weakly Compressible SPH
XSPH	Additional term for particle reordering

1. Introduction

1.1. Numerical background

Violent free-surface hydrodynamic flows exist in various industrial and research fields such as coastal and nuclear engineering. They include a diverse range of problems such as overturning or breaking waves and potentially explosive multi-phase pipe flow. Of particular importance is the interaction of the flows with various structures, such as coastal defences, tank walls or ship hulls. An accurate calculation of the potential impact forces generated by the violent flows is essential in designing a structure to maintain its integrity and avoid potential failure. The impact forces can be obtained either by an experiment or by a numerical simulation and predicting the behaviour, the velocity and density profile of the waves and especially their pressure field is a very important aspect of hydrodynamics. This study will focus on modelling water flows, but the research can be expanded to other Newtonian fluids such as oil, if their unique characteristics and attributes are taken into account.

The simulation of flows involving the interaction of different fluids is now commonplace. Understanding and predicting the motion of such multi-phase mixtures is of paramount importance for both natural and industrial phenomena.

The interface between two homogeneous fluids is one of the most important and interesting aspects of a multi-phase flow. A violent free surface occurs if the interface between the two fluids constantly changes, is not well defined and mixing between them occurs.

Using an experiment as a means of predicting fluid behaviour has the unquestionable advantage of being a real flow. In many cases however, conducting an experiment is either impossible or impractical, either because of the high cost (monetary or time allocation issues) involved with prototyping and testing, or because using a scale model does not allow for the observation of the more complex phenomena.

Using a numerical simulation for the prediction of a water flow, on the other hand, offers a more versatile option as virtually every flow can be simulated. It is also a less cost-intensive option depending on the hardware needed and the domain used can be of the same size as the real case, while there is explicit control over the input conditions. Setting up and analysing a simulation can be done in a relatively short time, as well as running multiple cases with different parameters. A wealth of experimental data is now also available for these flows, but the numerical models used to describe them are still lacking in comparison.

The major issue with any numerical simulation method is the dependence on the use of numerical models in order to simulate the flow behaviour. Computing complex flows by directly solving the governing equations is, in most cases, either impossible or extremely time-consuming, so the use of numerical models that simplify the mathematical process is necessary.

The numerical study of fluid flows is classified under the mantle of Computational Fluid Dynamics (CFD). This includes a wide range of methods that solve the governing Navier-Stokes equations as well as numerical models focusing on particular aspects of the fluid flow, such as turbulence or the viscosity. This particular study will focus on using meshless methods and in particular Smoothed Particle Hydrodynamics (SPH) in order to simulate air-water flows using novel hardware acceleration of graphics processing units (GPUs) as this allows for a more direct simulation of large cases with complex geometries and parameters, without setting up an expensive and time-consuming experimental installation.

1.2. Meshless Lagrangian Methods and Smoothed Particle Hydrodynamics

Implementations of CFD can be based either on an Eulerian or a Lagrangian description. In the Eulerian concept, the observer remains fixed in time as the flow is moving and the fluid quantities are functions of time and space. Any fluid variable f can be expressed at a fixed spatial point x as $f(x,t)$. This is the most common approach used in CFD.

Eulerian methods which encompass finite volumes, finite differences or finite elements are based on the concept of a mesh. A mesh is an arrangement of vertices, edges and faces used to map an object or domain. The flow quantities are calculated on the mesh node points which are fixed on space and time. Creating a mesh can be a complex, difficult and expensive task, especially when dealing with highly deformable flows or geometries with singularities.

For the flows simulated in the present study, the creation of a mesh would require a significant amount of time, sophisticated algorithms and computational resources (Sussman *et al.*, 1994). The free surface in violent air-water flows is rapidly changing, requiring a constant remeshing of the domain and due to the different state of each phase an explicit treatment of the surface is also needed (Belytschko *et al.*, 1998). In addition, the high velocities and dynamic pressure and density fields require a high resolution.

The other option for simulating a flow is the Lagrangian approach. Unlike the Eulerian one, the observer here is moving at the same velocity as the flow following each element as it moves in space and time. A flow variable f would then be expressed as a function of the starting point in space x_0 and the time variable as $f(x_0,t)$. It is not as widely used as the Eulerian approach, but interest has increased in the past 20 years.

Lagrangian methods are only now gaining traction because of the advancement of the available computational power and the decrease of the associated cost. Following a moving fluid element in time requires a large number of calculations increasing as the resolution increases, especially when calculating pairwise distances and forces. Only in the last few years has the hardware become capable of sustaining large Lagrangian simulations.

Lagrangian methods do not need the construction of a mesh. The fluid is instead represented as constantly moving node points, which resemble fluid volumes, such that they can represent a macroscopic flow. This makes them ideal for flows with large deformations, such as free-surface flows. An overview of the most prevalent Lagrangian methods will be given in Chapter 2.

1.3. Necessity for Multi-phase Modelling

In the majority of free-surface flows involving water, the air phase coexists next to the water surface. This occurs in most natural flows, including oceans and rivers. Additionally, a large number of industrial flows include interactions between air and water such as pipe flows and plunging waves. This study will focus on air-water flows at standard temperature; no heat exchange or phase changes will take place in the system.

In several of these cases, the effect of the air phase can be ignored, if the flow is characterised by low velocities and no or very little mixing occurs between the two phases. However, in cases where the flow becomes violent and highly transient, significant mixing between the two phases occurs and the effect of the air on the water flow cannot be ignored.

An example of the air affecting the water phase is the air entrainment, where an air volume is trapped within a water flow and moves with it. As a result it alters the velocity and pressure field of the water and, in case of an impact, the force exerted by the water. Modelling air entrainment is therefore vital when designing a coastal structure, a ship hull or a pipe. Another example is water aeration, whether that is for creating drinkable water, to increase the oxygen content in the flow or to treat sewage, removing foreign substances such as chlorine.

In this study, the focus will be on simulating violent flows with a highly transient and not clearly defined free surface where significant mixing occurs in the interface. The role of the air is then extremely important and it cannot be ignored. It is also an area where the numerical models have not yet reached the accuracy of an experiment as highly complex phenomena occur in the flow. Simulating both flows simultaneously presents then a considerable challenge.

The difficulty in simulating the air-water flows occurs from the vastly different properties of the two fluids as a result of them belonging to different states. At standard temperature and pressure, the difference in their densities is three orders of magnitude. There is also a large difference in their speeds of sound and the air is a highly compressible fluid as compared to the incompressible water.

The interface between them is usually defined by the surface tension present in the water, which maintains continuity of the interface for low velocity flows. Its effect however, is not as important in the cases studied in this report. If the necessary conditions are met, air water flows can also become turbulent and their chaotic nature makes these flows considerably more difficult to simulate.

The difference in their qualities makes the use of an identical numerical model for both phases unlikely. In practice, when using an Eulerian method, it is common to solve each phase separately, whilst treating the interface with a fraction method such as volume-of-fluid (VOF) or a curve tracking method such as the level-set algorithm. The mesh constructed in the interface (usually unstructured) would also need to be moving in order to capture the evolving interface.

Lagrangian methods on the other hand can represent different fluids simply as different sets of node points. They can use similar models for the two fluids, but extra terms in modelling the air phase or treating the interface are usually needed. These extra terms are easier to integrate in a code and their computation requires less time than constructing an unstructured, moving grid.

Multi-phase models on Lagrangian methods however, are still relatively new with interest of them increasing after the turn of the century. As a result, although research in this topic is increasing many issues still remain unsolved. One particular issue is the inability of running multi-phase cases with high resolutions due to the high computational runtimes required.

1.4. Parallel Programming and Hardware Acceleration

Running a simulation can be potentially time consuming depending on the complexity of the flow, the size of the domain and the duration of the case. Additional concerns are the resolution used for the computation and especially the numerical model used. As mentioned, Lagrangian models have a high computational time although they do not need the creation of a mesh.

The length of time required for a complete simulation is further increased when modelling the air phase. Not only do the qualities of air such as the high compressibility require a smaller time step but the size of the computational domain is increased with a finer resolution needed for the interface. For all these reasons a traditional serial computer code is usually insufficient to accurately simulate an air-water flow.

To improve this situation, the advancement of the computer science has allowed the creation of parallel programs, where the code is simultaneously executed by multiple processors. These communicate and exchange information with each other, handling only a fraction of the total data and greatly increasing the speed of the computation.

Massively parallel systems with hundreds or thousands of processors can also be used. They can significantly speed up the computation and handle an increased amount of data, allowing for finer resolutions. For Lagrangian methods there has been significant research in this field, with many codes being developed, greatly reducing the runtime required. However, there is not so far a code for massively parallel systems using a multi-phase model.

Graphics Processing Units (GPUs) have recently emerged as a viable tool for scientific computing due to their ability to process large quantities of data and their lower purchase and maintenance costs compared to conventional HPC clusters. Their massively parallel architecture is an ideal fit for the Lagrangian methods used in this study enabling the modelling of finer resolutions. The emergence of dedicated programming frameworks, such as CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) has also greatly reduced the complexity associated with their programming.

1.5. Study objectives

The main objective of this thesis is to develop a simulation program for air-water interaction in violent free-surface flows including a multi-phase model to increase the accuracy of the simulation. To achieve that, use of parallel programming and hardware acceleration using graphics processing units forms a core objective of this work; the result will be the reduction of the necessary computational time allowing not only an increase in particle resolution but realistic test cases to be performed.

Therefore, the objectives of this research can be summarised as:

- Implementation of a multi-phase model in an SPH GPU code
- Implementation of techniques for further reduction of computational runtime
- Optimisation of the GPU multi-phase execution algorithms
- Simulation of a variety of test cases in 2D and 3D with multiple resolutions to demonstrate the robustness of the code
- Comparison with the corresponding single-phase simulation and experimental data

1.6. Outline of the Thesis

This introductory chapter is immediately followed by a review of the published literature that will examine the foundations and the most recent advances in meshless and Lagrangian methods, focusing in particular in the SPH method. The publications regarding the advances in multi-phase modelling using SPH will be examined closely. Chapter 2 will also include an examination on the current state of parallel programming, especially regarding the use of GPUs as a means for accelerating the execution.

Chapter 3 will focus on the theoretical and mathematical background of the SPH method, with special attention being given to multi-phase schemes and especially the model chosen for the simulation of interfacial flows (Colagrossi and Landrini, 2003). The next chapter will examine the acceleration of SPH programs using GPUs, analysing issues and practices related to this approach as well as the GPU DualSPHysics code (Gomez-Gesteira *et al.*, 2012) used in the present study. The changes regarding the application of a multi-phase model as well as different methods regarding the optimisation of the code are documented. Runtime results regarding two cases and comparing the CPU and GPU versions of the code will be presented. The effects of the cases' extension to 3D on the runtime of the simulation will also be outlined.

In Chapter 5 compressibility issues in high resolutions and the treatment of the air particles is discussed and the use of a shifting algorithm (Xu *et al.*, 2009, Lind *et al.*, 2012b) is proposed as a solution. This chapter also presents a number of test cases used to validate the code and test its robustness. A dam break test case is compared to experiments and the corresponding single-phase simulation. The dam break is performed with both a dry (Koshizuka and Oka, 1996, Colagrossi and Landrini, 2003) and a wet bed where the movement of the gate is also included (Janosi *et al.*, 2004). The final 2D test case is a rolling tank, also compared to experiments (Botia-Vera *et al.*, 2010).

The accuracy of the code will also be tested for cases extending in the three dimensional domain. The violent impact of the water flow on an obstacle will then be investigated with the results being compared to both experimental data (Kleefsman *et al.*, 2005) and a single-phase SPH simulation (Crespo *et al.*, 2011a). The final chapter will then finish with a discussion of the results being presented and recommendations for future improvements and extension of the work.

2. Literature Review

2.1. Introduction

The aim of the present literature review is to assess recent advances in meshless methods, especially smoothed particle hydrodynamics (SPH). Different meshless methods will be reviewed and compared, followed by analysing the evolution of the SPH method and its application on fluid dynamics. Special attention will be given to the existing multi-phase methods for SPH and their applications in air-water flows. This investigation will also present the technologies used for parallel programming and program acceleration, in particular graphics processing units (GPUs) and their use with SPH.

2.2. Meshless and Lagrangian Methods

Numerical methods such as the finite element method are defined at distinct points connected by a computational grid or mesh. This is based on the Eulerian concept of the flow where the observer is stationary. The interaction between the different points of the grid allows us to define efficiently mathematical operators such as the derivative, which leads to an efficient computation of the governing equations (in the case of fluid dynamics the Navier-Stokes (NS) equations).

However, in a case where significant deformation can appear, such as a violent fluid flow or a simulation of a structural failure process where points in the mesh need to be destroyed, the grid needs to be constantly remeshed, which is a very expensive process (Belytschko *et al.*, 1996). In addition, treatment of discontinuities in a domain is also an issue, especially if their position and alignment changes (Nguyen *et al.*, 2008). A classic example is a free surface flow, where the border changes over time and can be extremely fragmented.

Meshless methods were created in order to address exactly these problems (Belytschko *et al.*, 1996). They are based on the Lagrangian concept of flow, where the observer and the flow are moving with the same velocity. The domain is entirely simulated with nodes or particles that are interpolation points for the fluid properties to be computed (Monaghan, 2005). The Lagrangian nature of these methods makes them ideal for the treatment of discontinuities due to their similarity with molecular dynamics (Hoover, 1998).

Meshless methods in their current form originated about 30 years ago when the vortex method (Chorin, 1973) and the smoothed particle hydrodynamics method (Gingold and Monaghan, 1977, Lucy, 1977) were introduced. However, research on this subject for the next 20 years was limited, until the advances in computer science allowed meshless methods to be used in practical applications, resulting in significant advances.

A summary of the main meshless methods will now be presented. We are interested in macroscopic applications, so methods such as Molecular Dynamics (MD), which are used in mesoscopic and microscopic applications are outside of the scope of this work, despite being meshless methods. The method used in the present study is smoothed particle hydrodynamics and the reasons will be explained later in the chapter.

- Vortex methods as they are currently used, were first proposed by Chorin (1973), (1978) and are used for the simulation of turbulent flows. Work on vortices however, has started much earlier with the calculation of Kelvin-Helmholtz instabilities (Rosenhead, 1931). The Navier-Stokes (NS) equations are considered in terms of the vorticity variable and discretised the vorticity instead of the velocity field (Barba *et al.*, 2005). The pressure is not explicitly solved but as the momentum equation is substituted by a vorticity transport equation by the application of the curl, the pressure is eliminated (Speziale, 1987). The velocity is then computed as an integral over the vorticity field, corresponding to the Biot-Savart law. The method proposed by Chorin was based on creating single point vortices. However, singularities can appear when the distance to the point vortex becomes very small, so the method is more often used with “blobs” with finite widths (Perlman, 1985).
  Simulating flows with the vortex methods became accessible after the creation of the fast multipole method (FMM) (Greengard and Rokhlin, 1987) which offered a fast solution to the N-body problem. Practical applications of the vortex method include flows around a cylinder (Koumoutsakos and Leonard, 1995, Smith and Stansby, 1988, Ploumhans and Winckelmans, 2000) a shock tube (Cottet *et al.*, 2000) or the penetration of a blade into a vortex core (Marshall and Grant, 1996). Acceleration of the simulation using GPUs has also been possible (Rossinelli *et al.*, 2010). The vortex method is most efficiently used with homogeneous infinite flows; its main issue however, is the lack of accuracy due to the distortion of the Lagrangian field in complex flows (Barba *et al.*, 2005).

- The Element-Free Galerkin (EFG) method was proposed (Belytschko *et al.*, 1994) as an improvement on the Diffuse Element Method (DEM) proposed 3 years earlier (Nayroles *et al.*, 1991). The DEM method replaces the FEM interpolation between the grid points with a Moving Least Squares local interpolation (Nguyen *et al.*, 2008). The EFG method enhances this approach using Lagrange Multipliers to impose the boundaries and is based on the full form of the derivatives instead of assuming them as constants (Belytschko *et al.*, 1994). Since it uses the weak form an auxiliary cell structure is used to define the quadrature points, so the method is not entirely meshless (Chen *et al.*, 2001).

  The main disadvantage of the EFG method is its high computational cost (Belytschko *et al.*, 1996). It has been extensively used for simulating discontinuities such as progressive cracking and fractures (Belytschko *et al.*, 1995, Lu *et al.*, 1994, Fleming *et al.*, 1997) or thin plate analysis (Krysl and Belytschko, 1995).

- The Reproducing Kernel Particle Method (RKPM) was originally introduced by Liu *et al.* (1995b). It originates from wavelet methods and attempts to correct the lack of consistency present in other meshless methods, especially their deficiencies close to the boundaries. The interpolation is similar to the SPH one, however, the function is treated as a window function and its coefficients are connected to the integral window transform (Liu and Chen, 1995). Scaling functions used to create wavelets are proposed as suitable candidates for the approximation. A correction function is used close to the boundaries (Liu *et al.*, 1995b).

  The RKPM has been used for structural dynamics (Liu *et al.*, 1995a) and large deformation analysis (Jun *et al.*, 1998, Chen *et al.*, 1997) and metal forming problems (Chen *et al.*, 1998). It has also been combined with an MLS algorithm (Liu *et al.*, 1997).

- The Meshless Local Petrov-Galerkin (MLPG) is a weak form method presented in 1998 (Atluri and Zhu, 1998). The weak form is computed locally eliminating the need for a background mesh as in EFG, while the Petrov-Galerkin method allows it to be simplified. The test function is not rigidly selected and different approaches can be used, leading to different MLPG schemes (Atluri and Shen, 2002). Cases solved with this approach include plate deformation (Qian *et al.*, 2004), convection-diffusion problems (Lin and Atluri, 2000) and the incompressible NS equations (Lin and Atluri, 2001). An issue with this technique is the difficulty in imposing boundaries (Atluri and Zhu, 2000).

- Dissipative Particle Dynamics (DPD) is a method derived from Molecular Dynamics (MD) modified in order to tackle issues with time and space scales applicable to hydrodynamic problems (Hoogerbrugge and Koelman, 1992). It was later adapted to ensure thermal equilibrium, as the earlier scheme was not conserving energy (Espanol and Warren, 1995). Particles move according to Newton's laws but the interparticle interactions allow for much greater scales.
  The DPD method can be parallelised using an MPI implementation (Sims and Martys, 2004). The method has been used in different fields such as hydrodynamics (Espanol, 1995), colloidal suspensions (Boek *et al.*, 1997) or modelling concrete flow (Sims and Martys, 2004).

- The Finite Pointset Method (FPM) was introduced in 1996 (Onate *et al.*, 1996a) and is based on point collocation for evaluating the hydrodynamic properties' derivatives. To construct the approximation functions a multitude of techniques can be used including Least Square approximation (LSQ), Weighted Least Square approximation (WLS) or Moving Least Squares (MLS). The technique has mainly been used for fluid mechanics problems such as modelling the compressible flow around an air foil (Onate *et al.*, 1996b) or free-surface problems (Vacondio and Mignosa, 2009). Multi-phase simulations have also been performed (Tiwari and Kuhnert, 2007).

- The Moving Particle Semi-implicit method (MPS) was introduced by Koshizuka and Oka (1996) for the simulation of incompressible free-surface flows. It is based on integral interpolation to approximate the strong form of the governing equations. Unlike SPH, however, the differential operator is based on a local weight averaging process rather than the gradient of a kernel function. The time scheme used is a semi-implicit prediction-correction process.
  A corrected MPS method has been proposed (Khayyer and Gotoh, 2008) and work has also been done in correcting the spurious pressure calculation that was one of the issues of this method (Khayyer and Gotoh, 2010a). The MPS method has been mainly

used in fluid problems such as breaking waves (Khayyer and Gotoh, 2008, Koshizuka *et al.*, 1998) but has also found use in other areas such as heat transfer (Zhang *et al.*, 2006).

- The Smoothed Particle Hydrodynamics (SPH) is one of the first meshless methods to be developed (Gingold and Monaghan, 1977, Lucy, 1977). It is based on integral interpolation of the variable properties and the continuous media appear as discrete points in the domain (particles). The particles' properties are then interpolated ("smoothed") over a predetermined distance using a kernel function. Contributions of neighbouring particles are weighted according to their distance.

  The method was initially introduced to tackle astrophysical phenomena such as star formation (Katz, 1992) and collision (Rasio and Shapiro, 1991) but was since adapted for solid mechanics and used for problems such as impact simulations (Benz and Asphaug, 1994), brittle solids (Gray *et al.*, 2001, Benz and Asphaug, 1995), metal forming (Bonet and Kulasegaram, 2000) and fragmentation (Randles and Libersky, 1996). It has also seen extensive use in fluid dynamics, especially in free-surface flows (Monaghan and Kos, 1999, Monaghan, 1994, Shao and Lo, 2003). Multi-phase models, which will be further discussed shortly, have also been developed (Colagrossi and Landrini, 2003) and an active effort is being undertaken to reduce the computational cost of the method, which is one of its drawbacks (Crespo *et al.*, 2011a). Recently, the SPH and MPS methods have been shown to be equivalent under certain conditions (Souto-Iglesias *et al.*, 2013).

In this study, the main interest lies in computing violent air-water flows. Flows of this kind are quite complex and include a free surface with numerous discontinuities. SPH creates a free surface for two interacting fluids directly and its particle-based scheme is ideal for the treatment of discontinuities (Monaghan, 2005) so it is a very attractive method for a large deformation analysis. In addition, a variety of models and corrections have been added to the classical implementation improving the accuracy and leading to a robust and mature method.

2.3. SPH for Fluid Dynamics

2.3.1. Classical SPH implementation

As mentioned before, SPH was developed in order to tackle astrophysical problems (Lucy, 1977, Gingold and Monaghan, 1977), in particular non-axisymmetric phenomena. It is still being widely used in the astrophysics field where large codes have been developed allowing simulations with hundreds of millions of particles (Springel, 2005). It was then quickly adapted with a scheme that conserved angular and linear momentum for fluid dynamics (Gingold and Monaghan, 1982).

SPH is a meshless particle method where the derivatives are found by integral interpolation (Monaghan, 1992). It allows the expression of the material properties such as density or pressure using local quantities at discrete Lagrangian locations. The interpolation is local at an area around a particle. Gradients can then be computed as the interaction between a pair of particles. A smoothing kernel is used to quantify the effect of the neighbouring particles. The method can be applied to both fluids and solids.

As a Lagrangian method, SPH is more suitable to simulate the large nonlinear deformations and the rapid flow movement than a traditional Eulerian method such as the finite volume method, which requires a very fine, adaptive mesh. This is ideal for fluid simulation and SPH has been used, as mentioned, extensively in this field to simulate both compressible and incompressible flows (Monaghan, 2005).

SPH, however, is not a method without issues. The classical SPH formulation described above suffers from accuracy and stability problems (Colagrossi and Landrini, 2003), in part due to the particle distortion (Koumoutsakos, 2005) and it does not have linear consistency near the boundaries (Belytschko *et al.*, 1996). In addition, applying the SPH to hydrodynamic problems requires the inclusion of boundaries, which are not present in astrophysics.

2.3.2. Boundary treatment

The truncation of the kernel near the boundaries is a serious problem for SPH. The value of any particle depends on the values of its neighbouring particles and since an interpolation is used, the number of particles is directly related to the accuracy of the computation. This reliance on the kernel however, becomes a problem when the particle is close to the boundary. As shown in Figure 2.1, for a particle close to the boundary, the kernel extends beyond the computational domain where no particles are present.

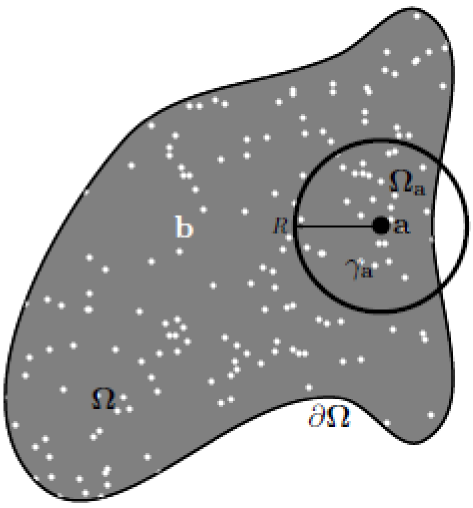


Figure 2.1: Kernel boundary interaction (Ferrand *et al.*, 2010)

Several methods have been proposed for addressing the boundary issue. A potential scheme is to extend the computational domain in order to include a number of fictitious boundary particles (Monaghan, 1994). These have the attributes (such as density and mass) of the fluid that is being modelled, but unlike the fluid particles, their motion or stationary character is prescribed. It has also been reported that exchanging density contributions among the fluid and boundary particles tends to assist capturing pressure peaks (Morris *et al.*, 1997).

This method, while simple, has a number of significant drawbacks. It needs a large number of additional particles to prevent particles exiting the domain and complete the kernel (Di Monaco *et al.*, 2009), increasing the computational time. As it will be discussed later, an important problem of this method is that the fluid particles tend to penetrate the wall and escape from the computational domain.

The large number of boundary particles however, means that the fluid particles near the boundary have more neighbours and the kernel support is not truncated (Morris *et al.*, 1997). The solution however, is now influenced by the values of the boundaries with the fluid particles next to the solid wall having a tendency to remain close to their initial positions, even if their neighbouring particles are moving (Monaghan and Kajtar, 2009a).

An alternative method is the use of repulsive forces based on Lennard-Jones potential (Monaghan, 1994). This method simulates the solid wall as particles which exert strong repulsive forces in order to prevent the fluid particles from exiting the domain. The force is applied along the normal direction to the wall and is dependent on the distance between the fluid and the boundary particle. The force describes the interaction between atoms and is used in molecular physics (Kulasegaram *et al.*, 2004). This method effectively eliminates the chance of particles escaping the domain, since the repulsive force will become infinite if the particle reaches the boundary points (Monaghan, 1994). It also has the advantage of being able to adapt to any geometry that is required. However, the interpolation is not completed with new particles so the inconsistency near the boundaries is not addressed and the repulsive force is not linked to the physical properties of the fluid and is dependent on empirical parameters (Marongiu *et al.*, 2007).

Another frequently used boundary treatment is the ghost particle method. The computational domain is extended by creating mirror particles of the existing fluid particles, which would have the same physical properties (Randles and Libersky, 1996, Libersky *et al.*, 1993). For a plane wall, this is similar to a symmetry condition. However, it is very difficult for complex geometries or for intersecting boundaries, such as corners. When simulating a curved wall for instance, a geometry transformation would be necessary, while when simulating a corner separate treatment of the faces is required (Le Touzé *et al.*, 2006).

Børve (2011) presented a generalised ghost particle method which generates a different number of ghost particles for each fluid particle. The method aims to keep the mass density in the kernel radius of a particle constant, regardless of the area of the kernel that is outside the domain by varying the mass of ghost particles.

In order to combine the advantages of the methods and to eliminate their disadvantages, hybrid approaches have also been proposed such as a hybrid of fictitious and ghost particles (Colagrossi *et al.*, 2009). In this approach, the fictitious particles are created in the beginning of the computation and an MLS interpolation method is used to obtain the necessary values.

Ferrari *et al.* (2009) proposed a hybrid method combining repulsive wall forces and mirror particles. In their approach, the fluid particles' density and pressure are affected by the mirror particles, which are not affected by the neighbouring fluid particles. An improvement to his work was proposed by (Vacondio *et al.*, 2012) who introduced a more complete support for the fluid particles. His work has been further improved in regards to its first and zeroth-order consistency (Fourtakas *et al.*, 2013b).

Kulasegaram *et al.* (2004) proposed a semi analytical approach which consists of renormalizing the density field with respect to the void area in the kernel. The momentum equation was then reworked (De Leffe *et al.*, 2009) and a semi-analytical scheme for computing the correction term was proposed (Feldman and Bonet, 2007). Finally, Ferrand *et al.* (2010) (2013) proposed a new time scheme to compute the renormalisation term and an extension for complex boundaries.

Overall, boundaries can be an awkward issue for SPH and solutions are still being investigated and developed.

2.3.3. Numerical stability corrections

To address the stability and accuracy issues, several corrections and models have been proposed. One of the earliest corrections proposed was by Lucy (1977) who used a bulk viscosity to prevent integration errors. That scheme however, did not conserve angular and linear momentum. A different scheme was then proposed (Monaghan and Gingold, 1983) using an artificial parameter to stabilise the algorithm, later revised to allow the simulation of high Mach number shocks (Monaghan, 1992). The scheme is similar to the von Neumann-Richtmyer artificial viscosity (von Neumann and Richtmyer, 1950) and introduces dissipation to prevent particles from penetrating each other (Monaghan, 1989).

The dissipation and the shear viscosity (Liu and Liu, 2003) introduced in the flow by this scheme can be detrimental to the accuracy of the results (Cha and Whitworth, 2003) but at the time of writing it is the most widely used viscosity correction for SPH (Liu and Liu, 2003). Monaghan has also proposed a different correction deriving from the dissipative terms of the Riemann solver (Monaghan, 1997), while a different laminar scheme has been proposed and tested for low Reynolds numbers (Morris *et al.*, 1997). Numerical simulation of the viscous term in the NS equations is also possible (Takeda *et al.*, 1994).

Regarding turbulent flows, discrete equations have been developed for both the mixing length model (Violeau and Issa, 2007a) and two-equation turbulence models including the k-ε (Violeau and Issa, 2007a) and the k-ω (Issa *et al.*, 2009) models. The k-ε model has been found to provide better velocity profiles than the mixing length model for a hydraulic jump (De Padova *et al.*, 2010). They will not be used in the current study, as modelling turbulence is beyond its scope.

Apart from the viscous term, there have been attempts to also correct the kernel and gradient functions. Following the kernel corrections in the RKPM method (Liu *et al.*, 1997), Bonet and Lok (1999) presented separate and mixed gradient and kernel corrections to the classical SPH approach, improving the accuracy near the boundaries. Their approach, however, did not conserve angular momentum.

An alternative approach for maintaining stability is the use of an approximate Riemann solution to determine the interaction between a pair of particles, especially the velocity and stresses in the contact surface (Vila, 1999). Using an Arbitrary Lagrange-Euler description, it was showed that this is a moving Riemann problem and using the equivalent steady Riemann problem it is possible to link the SPH method to the Finite Volume method. At this point however, the method ceases to be fully Lagrangian (Marongiu *et al.*, 2010).

A known issue with the SPH method is that particles under tensile stress have unstable motion. This issue is called tensile instability and can result in particle clumping. It is greatly dependent on the second derivative of the kernel function (Swegle *et al.*, 1995). Several schemes have been proposed for its correction including the use of specifically designed kernels (Morris, 1996), the use of additional stress points (Randles and Libersky, 2000, Dyka and Ingel, 1995) or the use of an artificial force (Monaghan, 2000, Gray *et al.*, 2001).

2.3.4. Weakly Compressible and Incompressible SPH

The approaches and schemes detailed so far are part of the so called weakly compressible SPH (WCSPH), in which incompressible flows are approximately treated as slightly compressible (Monaghan, 1992). The compressibility is enforced by the use of a low Mach number and a state equation is used to algebraically link the pressure and the density. The equation that has been widely used was proposed by Batchelor (1967).

The issue with the weakly compressible approach lies in the prediction of the density and pressure via the equation of state. A density renormalisation scheme utilising a zeroth order MLS function (a Shepard filter (Shepard, 1968)) is usually applied (Colagrossi and Landrini, 2003). This leads to a local averaging of particle densities and has insignificant effect on the accuracy of the results (Dalrymple and Rogers, 2006). The first order MLS correction can also be applied (Colagrossi and Landrini, 2003).

Schemes have also been proposed to treat the fluid as fully incompressible. The incompressible SPH schemes are based on the projection method (Chorin, 1968, Cummins and Rudman, 1999) and are using a Poisson equation in order to find the pressure values (Shao and Lo, 2003, Hu and Adams, 2007). Under this approach an intermediate velocity is projected in a divergence and curl-free domain. A different approach uses kinematic restrictions to impose incompressibility instead of the Poisson equation (Ellero *et al.*, 2007).

An improved scheme was proposed by Xu *et al.* (2009) who incorporated a shifting algorithm in order to correct the error caused by the irregular distribution of the particles and the kernel behaviour. The particles are slightly shifted in position and the hydrodynamic variables are updated using the Taylor series. The algorithm was improved by Lind *et al.* (2012b) who used a Fickian-based algorithm to compute the shifting position and incorporated a correction for the free surface. Skillen *et al.* (2013) also looked at the relationship of the shifting algorithm with the CFL time step condition.

2.4. Multi-phase Modelling with SPH

As mentioned, SPH, as a Lagrangian method can be effectively used in order to simulate violent flows where Eulerian methods can be difficult to apply, such as wave breaking or a potentially explosive multi-phase pipe flow. Most of these violent flows, however, are primarily defined by the interaction of multiple phases which need to be taken into account in a simulation.

One of the great advantages of SPH is that including more than one fluid is relatively straightforward, as it is possible to assign a separate set of particles to each phase with different equations of state (Monaghan, 2005). Handling the interactions among them however, is greatly dependent on their phase and the ratio of their hydrodynamic properties such as the density. Treatment of the interface may be necessary but compared to the Eulerian methods surface tracking is relatively easy with no need for a finer resolution. The increased number of particles however, has a significant impact on the computational runtime. This study will focus on simulating flows between two different fluids. Simulating solids and their interaction with fluids is beyond the scope of this research.

Extensive research has already been conducted in the field of multi-phase modelling with SPH. Schemes have been proposed for both liquid-gas and liquid-liquid flows. The first research publicised, was, to the best of the author's knowledge, a model regarding dusty gases that was published in 1995 (Monaghan and Kocharyan, 1995).

The first publication regarding the interaction of multiple fluids simulated gravity currents flowing in a ramp (Monaghan *et al.*, 1999) which used the classical SPH formulation, treating each fluid as a separate set of particles, without any corrections at the interface. Results were satisfactory but the density ratio between the fluids (water and saltwater) was very small (never larger than 1.15). This scheme is not suitable for fluids with large density ratios (fluid-gas flows) as several instabilities in the area between the two substances are developed due to the high density gradient at the interface (Colagrossi and Landrini, 2003).

Simulation of the interactions between fluids with high density ratios was investigated by Nugent and Posch (2000). The Van der Waals (vdW) pressure equation was considered and the interatomic forces were translated to an attractive force among the particles. This force acts at the interface due to the high density gradient but is negligible in the bulk of the fluid. The authors note that instabilities may appear in the interface and propose the use of a larger smoothing length in order to eliminate this issue.

A similar method was also used by Colagrossi and Landrini when they proposed a new scheme for the computation of interfacial flows. The use of a similar attractive force in the lighter fluid was suggested to prevent particle dispersion at the interface (Colagrossi and Landrini, 2003). A new form of the SPH approximation and the viscous term was proposed for free-surface flows. The equation for the conservation of density used a different formulation which is non-conservative (Hu and Adams, 2006). A density re-initialisation term was also used.

For the different fluids, the same equation of state is used (Batchelor, 1967), modified for the lighter fluid. Due to the density ratio the speed of sound used for the lighter fluid in this equation needs to be significantly larger. This leads in turn to a significantly smaller time step as the stability of the method is linked to the speed of sound. This method was applied to a dam break case with air-water flow and the results were in good agreement with other numerical methods.

A different approach was followed by Hu and Adams (2006). The density is not affected by the volumes of neighbouring particles, but only the particle's own volume. The particles' specific volume is the primary hydrodynamic variable in this approach, replacing the density in the new discrete SPH equations proposed. For the viscous term, instead of the artificial viscosity (Monaghan and Gingold, 1983) an approximation of the inter-particle shear stress is used. A surface tension term using a colour function is also included.

Test cases examined included a capillary wave and a Couette flow. The model appeared to be in good agreement with both experimental results and analytical solutions even in cases with large density ratios. The main disadvantage is the complexity of the model. Estimating the local curvature and the normal to the surface is computationally costly and may lead to errors when the surface is not well defined, such as a free-surface flow (Tartakovsky *et al.*, 2009, Morris, 2000). Boundaries could also be a significant issue if the geometry complexity is increased.

The previous work was followed by the proposition of a model for incompressible fluids (Hu and Adams, 2007) using the projection formulation similar to those described in section 2.3.4, introducing an intermediate time step for the velocities and the position. These values are then modified at the full time step to ensure the zero-density-variation and the velocity-divergence-free condition. For the computations at the interface, averaged hydrodynamic values leading to von Neumann boundary conditions are used (Cummins and Rudman, 1999).

The incompressible model was improved in 2009, when the authors introduced a constant density approach (Hu and Adams, 2009) in order to model flows with high density ratios. As with the previous model complexity, the computational resources needed and the flow at the free surface are serious issues (Grenier *et al.*, 2009).

Grenier *et al.* (2009) proposed a model that combines the specific volume approach of Hu and Adams with the Colagrossi and Landrini scheme. The same equation is used for both phases and an additional repulsive force is used with the pressure gradient. They also propose a different density re-initialisation scheme. A surface tension term based on a colour function and a viscous term (Flekkoy *et al.*, 2000) modified to conserve angular momentum (Monaghan, 2005) are used. The method is used to simulate an air bubble rising in water and gravity currents.

Another approach considered was based on a SPH – finite volume hybrid, different to previous models (Leduc *et al.*, 2009). More specifically, an ALE description with a conservative form of the Euler equations is used so the inter-particle interaction is a one-dimensional Riemann problem. If the particles belong to different fluids a correction is used for the lighter fluid. Two surface tension models were considered: the Continuum Surface Force (CSF) model (Brackbill *et al.*, 1992) and the Local Laplace Pressure Correction (LLPC). Results showed that the LLPC model is more accurate, while the CSF model depends on predicting the curvature of the surface (Rogers *et al.*, 2009).

The method was later improved using a preconditioned Riemann solver (Leduc *et al.*, 2010). The preconditioning was used to treat the artificial diffusion caused by the ALE formulation in the previous model, especially near the interface. The method was tested with the dam break test case and the results were improved compared to the previous model, but more testing is needed.

An approach with an additional repulsion term for the lighter fluid was also proposed by Monaghan (2011) later improved by Monaghan and Rafiee (2013). The new term substitutes the artificial viscosity term (Monaghan and Pongracic, 1985) and is dependent on the density ratio between the fluids. Similar to the methods presented previously (Colagrossi and Landrini, 2003, Grenier *et al.*, 2009), this algorithm uses a higher speed of sound for the fluid with the lower density but its actual value is smaller allowing for a larger time step. The major issue with this approach is its empirical nature and the need for a different configuration for each problem.

Shao (2012) presented a new method for the treatment of incompressible fluids. In this approach two methods are proposed, with two different ways of treating the interface. The first model is coupled; the SPH equations do not separate the fluid phases and in order to treat the interface, a standard ISPH treatment is used (Shao and Lo, 2003). For the second model, the fluids are decoupled, meaning that each phase is treated separately and they interact with each other by exchanging shear stress and pressure.

A combination of an incompressible and a weakly-compressible method was presented by Lind *et al.* This approach has been used in air-water flows, where the air is modelled as a compressible gas, while the water uses an incompressible approach (Lind *et al.*, 2012a, Lind *et al.*, 2013). To couple these two methods, the incompressible phase provides a velocity boundary condition while the compressible gas creates a pressure boundary condition for the liquid. Separate time step sizes are employed in each phase, with the incompressible time step ten times larger than the compressible one (Lee *et al.*, 2008).

In this study, a gas phase needs to be simulated. In that regard, only the multi-phase models that are capable of simulating a compressible phase will be considered. A short comparison of the main multi-phase models for an air-water SPH simulation can be seen in Table 1.

Simulation of interfacial flows (Nugent and Posch, 2000), (Colagrossi and Landrini, 2003)	+ Successfully simulates violent air-water flows + Easy to parallelise on a GPU - Dependence on empirical parameters - Very small time step required for stability
Macroscopic and mesoscopic flows (Hu and Adams, 2006, 2007, 2009)	+ Conserves linear momentum + Successfully simulates violent air-water flows + Extended to incompressible SPH - Difficult implementation on a GPU - Expensive treatment of the free surface
Hamiltonian interface SPH simulation (Grenier *et al.*, 2009)	+ Quick convergence + Conserves linear momentum - Difficult implementation on a GPU - Dependence on empirical parameters
Lagrangian particle model (Tartakovsky *et al.*, 2009)	+ Easy to parallelise on a GPU - Dependence on empirical parameters - Untested for violent flows
Multi-phase flow using acoustic Riemann solver (Leduc *et al.*, 2009, 2010)	+ Successfully simulates violent air-water flows + Explicit treatment of the free surface - Requires the creation a Riemann solver - Difficult implementation on a GPU
Repulsive term model (Monaghan, 2011, Monaghan and Rafiee, 2013)	+ Easy to parallelise on a GPU - Dependence on empirical parameters - Issues with high density ratios
Combination of WCSPH and ISPH (Lind *et al.*, 2012a, 2013)	+ Successfully simulates violent air-water flows + Accurate modelling of the pressure field - Requires an incompressible solver - Difficult implementation on a GPU

Table 1: Comparison of different multi-phase models for an SPH simulation

2.5. Parallel Programming

SPH is a modelling method with extremely high computing requirements. This is due to the large number of calculations needed to simulate the interactions between the particles (Viccione *et al.*, 2008). Computing power is still not sufficient for SPH simulations to run quickly, especially when compared to methods such as finite elements or finite volumes.

In order to achieve sufficient accuracy with SPH, a large number of particles is needed increasing the necessary time for the completion of the simulation. Considering that real cases need potentially millions of particles to be modelled, the conventional computational time for such cases can be prohibitive (Crespo *et al.*, 2011a). This locks us in a vicious cycle where either accuracy or time need to be sacrificed in order to complete the computation.

This is especially true with a multi-phase SPH program. The time step for the lighter fluid can be significantly smaller than the step for the other phase (Colagrossi and Landrini, 2003). It will be discussed later in this review, that paradoxically under certain conditions a gas phase can potentially require a time step that is two orders of magnitude smaller, exponentially increasing the computational runtime.

Also, when modelling a multi-phase case the number of particles has naturally increased compared to the single-phase case as an additional area of the domain needs to be modelled. In certain cases, where multiple spatial scales are present such as coastal engineering problems (Dominguez *et al.*, 2013a) the particle number can increase multiple times. The increase is even larger if we are modelling the case in three dimensions as the number of particles increases, even without changing the resolution.

Therefore, if modelling of large cases is to be possible hardware acceleration is necessary (Crespo *et al.*, 2011a). In order to overcome the hardware limitations, we can use parallel computing. In the past, executing the code could only be done in serial sequence, i.e. for an instruction to be executed, all the previous ones had to have been completed. With parallel programming, however, program sections are given to separate nodes and executed simultaneously, significantly reducing the computational cost. Using a parallel approach is ideal for a Lagrangian method such as SPH. Different nodes can be used to update the values of different particles in the domain simultaneously.

2.5.1. Using High Performance Computing

The first possible step in incorporating some form of parallelism is to fully use the cores and the memory available in a system. Most computer processors today possess some degree of parallelism with the system memory available to each core simultaneously, so with parallel computing the processing power of each core can be assigned to creating multiple threads executing the code in parallel. The data are then uploaded in the shared memory so they can be accessed and modified by each thread.

OpenMP (Open Multi-Processing) is a set of compiler directives and library routines that allow programming languages to take advantage of shared memory architecture (Dagum and Menon, 1998). In a program using OpenMP there are two different sections, a serial and a parallel one. The serial code is executed consecutively in a master thread while the parallel code divides the master thread into a number of slave threads which are then assigned to different processors. Each thread is executed separately from each other and when the parallelised code ends, the threads return to the main thread which executes the rest of the program (synchronisation).

When running an OpenMP code memory can be either shared, meaning that every thread has access to it, or thread-private. The latter is useful when a thread is changing a specific set of data (Clark, 1998). Using this method allows for an easy implementation for parallelism in an existing code but it is functionally limited due to the need for shared memory (Vajda, 2011). The shared memory can create extensive synchronisation and data overwriting issues when a large number of processors are accessing it, while flow dependency in the code can render some parts impossible to execute in parallel (Suess and Leopold, 2008).

The next step to parallelism is to create a massively parallel system by connecting several processors through a network. This allows us to use thousands of cores and to decrease the computational time even further. MPI (Message Passing Interface) is a communications protocol that supports point-to-point as well as collective communication among computers and has already been used for simulating SPH applications (Rabczuk and Eibl, 2003, Ferrari *et al.*, 2009, Guo *et al.*, 2013). Each computation consists of a number of processes which are launched at the beginning. Each process is running separately, running its own copy of the process with access to a local memory. The processes can exchange information by using the MPI subroutines (Gropp *et al.*, 1994, Guiffaut and Mahdjoubi, 2001).

The logical extension is to interconnect systems with hundreds of thousands of processors using high speed protocols such as infiniband, creating a supercomputer. They are placed in close proximity to each other (e.g. in a computer cluster), saving considerable time transferring data and making it possible for the processors to synchronise their work. Another approach that is slowly gaining traction is distributed computing, where discrete machines are connected via a network; each machine executes a number of smaller tasks that are then uploaded in the network and integrated in the main solution.

It is important to note that the speed-up of the parallelisation is not linear. Using twice as many processors will not halve the computational time due to limited memory bandwidth as well as Amdahl’s law, which dictates that the speed-up of a program is dependent on the portion of the program which can be parallelised (Amdahl, 1967).

While using a massively parallel CPU system will significantly decrease the computational time, its cost can be prohibitive. Obtaining the processors needed, as well as sufficient space to store them is extremely expensive. The cost of connecting the machines and their maintenance costs, especially the energy and the cooling required are also quite significant (McIntosh-Smith *et al.*, 2012).

Another limiting factor is the interaction among the computers. Even if they are connected with the largest bandwidth possible (which naturally increases the cost), the communications with each other as well as the memory create limits to the speed of the computation. Indeed, modern processors can usually compute data faster than they can be transferred, while there is also the matter of synchronisation.

Creating a massively parallel CPU cluster, while effective, is a costly investment with large energy requirements. For this reason, the Graphic Processing Units (GPUs) have recently emerged as an alternative viable programming medium. They have a massively parallel architecture, which is highly suitable for a Lagrangian method.

2.5.2. Using Graphics Processing Units for Hardware Acceleration

GPUs are specialised hardware designed to process large quantities of data to be displayed on computer screens in real time. Primarily used for video games and image processing applications, their massively parallel architecture means they have emerged as an attractive tool for scientific computing. Along with dedicated programming language framework, such as CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language), general purpose graphics processing units (GPGPUs) have recently emerged as a viable alternative to HPC due to lower purchase and maintenance costs (Herault *et al.*, 2010).

The unique architecture of GPUs makes them particularly suitable for computationally intensive simulations using Lagrangian methods such as SPH. The large number of multi-processors on a GPU enables speed-ups close to two orders of magnitude compared to a single CPU code (Crespo *et al.*, 2011a). A problem with GPUs is that they are not as suitable for data-intensive applications. The data must fit on the memory of the GPU which has currently, on the high-end cards, an upper value of 6GB which is much smaller than the CPU RAM (as well as more expensive) and cannot be upgraded.

A major breakthrough in the use of GPUs as a computing tool came in 2007 with the release of the CUDA programming language. Developed by nVidia, CUDA did not use rendering operations to model mathematical equations but functioned as an extension to the C/C++ language handling the communication and data transfer from the CPU to the GPU and vice versa. The open industry standard OpenCL, which was released in late 2008, also followed this approach.

However, at present OpenCL is less suitable for scientific programming as CUDA is a better optimised and developed choice to further reduce the computational time. Moreover, a large number of libraries and algorithms have been ported to CUDA, while third-party wrappers for other programming languages have been released allowing for greater flexibility. Therefore, this project uses the CUDA language as a means of communication between the CPU and GPU.

2.6. GPU Acceleration with SPH

2.6.1. Texture rendering

The first work on using GPUs to accelerate SPH was performed by Amada *et al.* (2004). Their program was developed before the release of CUDA and OpenCL, so they used the OpenGL texture rendering approach. Their program was not coded exclusively on the GPU;

the initialisation of the variables as well as the creation of a neighbour map of the particles took place on the CPU and were transferred to the GPU.

In order to transfer the neighbour map on the GPU, a texture map structure was used which also contained the attributes of each particle. The SPH equations are then executed on the GPU by using this texture map as a reference. In order to handle the particle collision, they also created a triangle mesh using the attribute and boundary texture.

Simulations were conducted using the CPU-GPU approach and were compared to a CPU-only approach. The GPU was in general about 2 times faster than the CPU. A small number of particles were however used for their simulations and a larger speed-up was expected when the particle resolution is increased. Speed-up can also be increased if a larger portion of the program was parallelised and sent to the GPU.

The next simulation on a GPU was by Kolb and Cuntz (2005). They also used texture mapping in order to store the necessary data for the computation. Unlike the previous work however, the whole computation is performed on the GPU. The CPU was only used for loading and storing the variables.

In order to map the particles 5 texture arrays were created with two of the arrays used to map the velocities and two more for the particle position. The last array stored the particle constants. The authors found that the output and input arrays for the computation should be different, so for the position and velocities one array was acting as input and the other as output with their role reversed in each time step.

The authors also performed a three dimensional simulation using the texture rendering of the graphics card. The texture output in this graphics was two dimensional so the authors divided the three dimensional computation into a series of two dimensional slices. Communication among the slices was performed using a trilinear approximation scheme, which introduced an interpolation error in the computation.

The first major computation on SPH using GPUs was performed by Harada *et al.* in 2007, shortly before the CUDA compiler was released (Harada *et al.*, 2007b, Harada *et al.*, 2007a). They used the texture format to store the particle attributes, constructing the data texture exclusively in the GPU. Hence, the memory transfer to the CPU which would significantly slow down the computation was avoided. In their simulation, great attention was given in producing the same computational burden for each processor. For this reason, the computation region is divided in slices, which have one dimension less than the domain. A dynamic grid was then developed for locating the particles, which the authors found to be the most efficient way to use memory.

The authors performed computations with different particle numbers and found that the GPU had a consistently lower runtime than the CPU. They found that as the particle spacing is decreasing the discrepancy between the two versions is increasing. It was also shown that dynamic allocation of the particles is more computationally efficient than a rigid grid for a sliced data structure and large particle numbers.

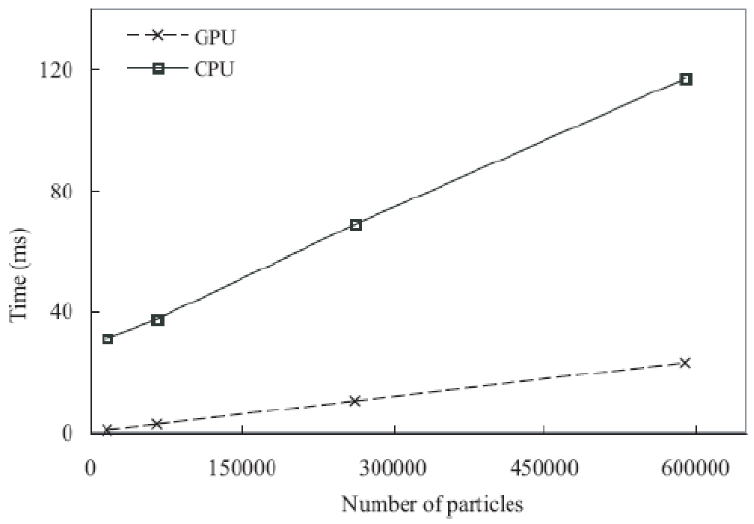


Figure 2.2:Comparison of construction times of the sliced data structure in the CPU and the GPU (Harada *et al.*, 2007a)

In the latter model, Harada *et al.* used a three-dimensional grid for the creation of a neighbour list, reducing the cost of searching for neighbouring particles. Since GPUs use a 2-D texture, the neighbour list was again divided into a series of slices placed in texture memory. Data was exclusively stored in the GPU memory to speed up the computation. With these improvements, Harada *et al.* reported that the computation on the GPU was 28 times faster than the computation on the CPU for 260,000 particles (Harada *et al.*, 2007b).

2.6.2. CUDA Programming Extension

The release of the CUDA compiler made GPU simulations much easier, offering a greater chance to improve the computational speed. Crespo *et al.* (2009) and Hérault *et al.* (2010) presented a comparison of a GPU with a CPU code. In order to take advantage of the CUDA architecture and optimise the program the authors followed these guidelines:

- The use of memory is optimised, by loading data from the shared GPU memory instead of the global memory whenever possible.
- The data transfers to and from the CPU are minimised.
- Only the variables necessary for solving the equations are copied to the GPU memory.
- Variables are stored sequentially, reducing the number of transfers.
- Constants are saved in a subset of the global memory called constant memory, where the access is faster.

The computation of a logical function in a GPU is much slower than a numerical function. Therefore, an effort to eliminate these functions and replace them with equivalent numerical statements was made. Extensive use of reduction operations to reduce the number and the size of the variables was also being performed.

Crespo *et al.* (2009)compared several versions of the GPU code with a CPU only approach and found that between the fastest versions of the code the GPU had a speed-up of about 11-12. The use of a neighbour list also increased the computational time considerably in both approaches. A large cause of slowdown for the GPU code was the copying of the memory between the host and the GPU device.

Hérault *et al.* (2010) developed an SPH program using the CUDA programming language, called GPU-SPH, by developing the existing SPHysics code. They also found that in order to have the highest performance on GPUs, data stored in the shared memory must be maximised and the calls to the global memory must be minimised. They also found that the position of data in the memory had an effect on the performance. Storing data required by the same processor in memory addresses close together offered a better performance.

Access to the global memory has a latency compared to the share memory. In order to minimise this latency and prevent the threads from remaining idle, each thread processes only one particle and the functions are designed to perform independent numerical instructions while waiting for the memory access to be completed. Conditional instructions were also minimised, not only in order to improve the performance, but also to avoid overwriting data due to the lack of synchronisation of the processors (Herault *et al.*, 2010).

Running the code on several GPUs reveals that the neighbour list as well as the time step is greatly dependent on the memory, since performing the mathematical arguments requires the loading of particle data from the memory. In every case, the GPU code shows an improvement in performance over the CPU code. However, the compared CPU code is not optimised and the GPU speed-up gained is expected to be reduced by half when the CPU code is optimised.

Oger *et al.* (2010) proposed the acceleration of an existing parallel CPU code by translating a part of it to GPU code. This strategy means that the overall speed-up will be limited by Amdahl’s law (Amdahl, 1967). To prevent data access conflicts and to reduce the possibility of computational errors, the authors also proposed that each thread will compute only one particle. They note however, that this method may cause latency due to the discontinuous memory access.

The authors implemented a method for particle sorting based on Peano-Hilbert space filling curves. They will allow the projection of the multi-dimensional data to a one-dimensional curve, which will give the most efficient positioning of the particle data in the memory, so that the access latency is minimised.

The hybrid scheme proposed by Oger *et al.* is focused on the interaction among particles and specifically at the momentum and continuity equations since it is the main cost of the computation. This part of the code is sent to the GPU connected to the CPU with CUDA memory directives avoiding the need to rewrite CPU code, while the CPUs themselves communicate using parallel directives (Oger *et al.*, 2010). Each CPU can host a different GPU allowing for the use of multiple GPUs in a single computation.

The authors tested the SPHERIC dam break benchmark and found that for a single precision code with 155,000 particles, the speed-up achieved is 1.86 (Oger *et al.*, 2010). For the double precision code the speed-up is reduced and is equal to 1.1 overall. These results clearly show that in order for the speed-up to be important the entire code needs to be translated to the GPU.

Following upon their previous work, Crespo *et al.* (2010) presented a dual CPU-GPU code, DualSPHysics, which is based on the existing SPHysics code. This code can be run as a CPU-only code or as a GPU code controlled by a CPU part using both C++ and CUDA to merge the implementation of the CPU and the GPU code. Parts of the code are shared between the two versions making it easier to compare the computational time.

The authors used a dynamic memory system to create the neighbour list and optimise the data transfer and also found that the format of the output files affects the computation. For one million particles setting the output format to binary files instead of ASCII will give a 20% increase in computational runtime.

This code is currently being improved (Crespo *et al.*, 2011a, Crespo *et al.*, 2011b). A cell-linked system has been introduced as well as a reduction algorithm for the calculation of the new time step. The radix sort algorithm for CUDA is used for the creation of the neighbour list (Dominguez *et al.*, 2011). The DualSPHysics code will be analysed further in the next chapter.

As mentioned before the entire particle data set needs to be loaded to the GPU memory because loading main computer RAM is one or two orders of magnitude slower. Valdez-Balderas *et al.* (2011) have proposed a massively parallel SPH scheme that will allow the use of multiple GPUs in a SPH computation.

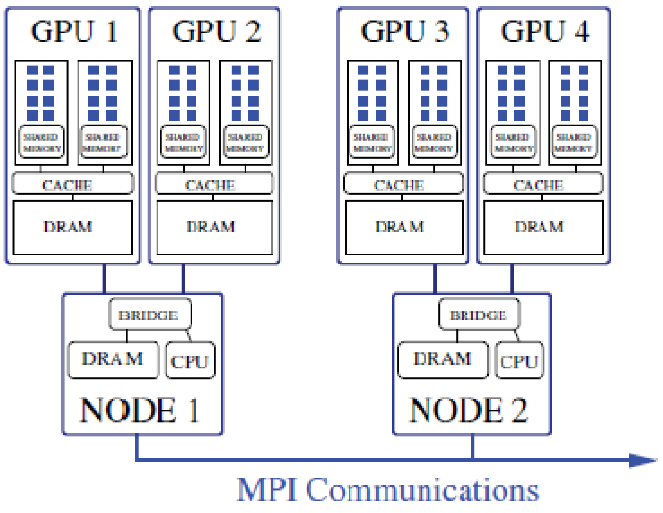


Figure 2.3:Illustration of a multi-GPU system with two nodes, each of them hosting two GPUs (Valdez-Balderas *et al.*, 2011)

The scheme proposed is based on the DualSPHysics code, which is extended with MPI directives and a ‘volume domain decomposition scheme’ in order to allow communication among the GPUs as illustrated in Figure 2.3. A different portion of the computational domain is assigned to each GPU which then analyses every particle in its domain. This approach is different to that of Oger *et al.* (2010) in that a more low-level approach is being used since the program is written in CUDA instead of applying directives to the GPU.

At each time step, particle data needs to be exchanged between the different GPUs. Similar to other parallel SPH implementations each sub-domain has a ‘halo’, an area within 2*h* distance from the neighbouring sub-domain boundary (Valdez-Balderas *et al.*, 2011). To update the particles in the sub-domain area, the radix sort algorithm is used. When looking at the results, the authors found that the cost of the CPU communication is the main cause of latency and is naturally increasing as the number of nodes is increasing.

The MPI-CUDA code has recently been improved and simulations at a large scale have been possible. Dominguez *et al.* (2013a) presented a simulation of an oil rig with over a billion particles. The simulation was possible due to the introduction of dynamic load balancing (Dominguez *et al.,* 2013b) in the code and the improvement of the communication and data exchange among the CPU and GPU systems.

2.7. Applications

2.7.1. General Application

Using a multi-phase model represents a significant investment of computational resources and a large number of alterations to the code, in addition to its need for parallel programming and hardware acceleration. To justify its use over the single-phase model, it needs to be applied to natural and industrial flows that involve significant interactions between the two phases, whether that are mixing, air entrainment or simply the alterations in the pressure and density field occurring by the interactions.

For natural flows, interactions between the water and the air occur in any water body. Most of these interactions such as standing waves occur for very low velocities, where the air phase does not affect the water significantly and can therefore be modelled with a single-phase model. If the velocities however increase and the interaction becomes violent, the air phase has a significant role in the flow evolution (Grenier *et al.,* 2009).

2.7.2. Breaking Waves

A wave reaching a critical point in its amplitude and subsequently transforming large amounts of wave energy in turbulent kinetic energy is called a breaking wave. Four types of breaking waves exist (Wiegel, 1992): spilling, plunging collapsing and surging breakers.

Breaking waves can occur in any body of water and in any position as long as the wave amplitude is sufficient. They mostly appear in shallow water areas such as beaches as their amplitude increases because of the interaction with the ocean floor (Peregrine, 1967). They are highly non-linear phenomena and finding an analytical solution is possible only for idealised conditions (Lin *et al.*, 1999).

All breaking waves include some degree of mixing with the air phase. Visually, this is indicated by the appearance of foam, which is formed by trapped air pockets inside the water flow. The least mixing occurs in the surging waves, while the plunging breakers and to a lesser degree the collapsing breakers can trap large pockets of air when the crest falls forward. The entrained air can remain inside the water flow for a considerable amount of time.

2.7.3. Wave Impact and Overtopping

The main reason for modelling breaking waves is their interaction with the shoreline and the offshore structures. When reaching the shore the still water level is increased. The vertical increase is called the wave run-up and determines the area that is affected by the wave (Kobayashi, 1997). The prediction of the run-up is essential for predicting sediment transport and preventing shore erosion (Masselink and Puleo, 2006). For this study however, the most interesting aspect of the run-up is the interaction with - and the potential overtopping - of coastal structures.

Overtopping over coastal structures can cause serious property damage as well as endanger human lives. It is a very complex turbulent flow involving numerous phenomena including wave shoaling and wave breaking. Reflected waves and the onshore wind conditions may also have an effect (Hu *et al.*, 2000). Experimental results are limited to flume experiments which cannot cover the full range of the phenomenon, so research has focused on numerical models (Titov and Synolakis, 1998, Hubbard and Dodd, 2002).

SPH is a potentially powerful tool for simulating structure overtopping due to its ability to track the free surface, while it can deal with vorticity and turbulence (Dalrymple and Rogers, 2006). The multi-phase model can be used to further enhance the accuracy of the simulation by taking into account the influence of the wind conditions and, in the case of breaking waves, predict the effect of air entrapment.

Overtopping is not only an issue for coastal structures but for ships and offshore installations as well. If the height of the waves is large enough the water can flow on the deck damaging equipment and endangering the crew. The phenomenon is named green water loading (Buchner, 1996). Using a single-phase SPH simulation (Gomez-Gesteira and Dalrymple, 2004) and (Dalrymple and Rogers, 2006) have modelled the overtopping of a flat plate.

Another aspect of the wave-structure interaction is the impact force of the wave on the structure, particularly important when designing coastal defence structures. The effect can be modelled in a laboratory, however, the flow becomes especially complex when the air is mixed with the water flow (Peregrine and Thais, 1996). The air is of particular importance because of its compressibility. At small scales, the air phase has relatively little compression while in large-scale application the pressure can reach several atmospheres, causing major reductions in the entrained air volumes (Bullock *et al.*, 2004).

Numerical methods have been widely used to simulate the impact force, including SPH (Gomez-Gesteira and Dalrymple, 2004, Crespo *et al.*, 2011a). The air phase has been simulated mainly using the VOF method to track the free surface (Wemmenhove, 2008) or a pressure-impulse model (Wood *et al.*, 2000). The research field is not saturated; a massively parallel application of a meshless multi-phase model will facilitate a simulation with high resolution while treating the air as a compressible fluid allowing for the complex phenomena present to be represented numerically.

The impact force greatly affects offshore structures since they are constantly subjected to waves impacting on the supporting structure. The impact depends on the weather on this location and it can be particularly high in cases such as storms (Kleefsman *et al.*, 2005). The structures are in fact designed on the principle of the hundred-year wave, a statistically created wave which may appear at a certain location once per century (Holthuijsen, 2007). The concern for the offshore structures does not only apply to instantaneous impact but also to the fatigue from the oscillatory load by the ocean waves (Wirsching, 1984).

However, the greatest impact forces and the greatest property damages occur by the tsunami waves. Tsunamis are earthquake-generated sea waves that, when propagating near the shoreline, display significant height increase due to the offshore bathymetry. The resulting wave can travel inland for significant distances causing great material damage and potentially endangering human lives. Research focus has been shifted to predicting their run-up near the shore in order to more effectively design appropriate coastal defences and mitigate potential damage (Li and Raichlen, 2002).

An interesting phenomenon appearing in breaking waves that can also potentially generate large pressures, albeit only in a local scale, is the flip-through. The wave is impacting at a vertical wall but the air is not necessarily trapped (Lugni *et al.*, 2006). The phenomenon is characterised by a rapid rise in the water level near the wall; a vertical jet is being formed and is rising upwards with such speed that prevents the face of the wave from impacting with the wall directly (Peregrine and Cooker, 1990, Bredmose *et al.*, 2010). Despite the lack of direct impact the resulting flow can create high pressures in the wall region with accelerations up to 2000*g* (Peregrine and Cooker, 1990).

2.7.4. Industrial Applications Including Sloshing

Violent air-water flows do not only appear in large-scale flows occurring at large water bodies such as the ocean. They are a common occurrence in any flow involving air and water regardless of their volumes. They are also present in several industrial applications, including the flow inside half-filled tanks and pipes and mixed air-water flows.

Sloshing refers to the movement of a liquid inside another object, which is also under motion. The slosh is affected by the depth of the liquid in the vessel, as well as the size of the vessel and the type of movement it is subjected to. The sloshing effect changes the weight distribution of the liquid and the forces exerted on the wall. This case has several applications in modern engineering including ships and trucks transporting liquids (oil and gasoline in particular) (Faltinsen *et al.*, 2000) as well as aircraft and spacecraft tanks (Reyhanoglu, 2003). At a smaller scale, sloshing effects are present in every fuel tank found in an automobile.

Of particular interest is also the gas–liquid flow inside a pipe. The form this flow takes depends on a number of different parameters, such as flow velocity and temperature, the buoyancy of the gas bubbles and the shape and alignment of the pipe (Nicklin, 1962). The flow can be however categorised using different flow patterns depending on whether the pipe is vertical or horizontal (Hewitt, 1982).

Regardless of the size of the gas bubbles, in vertical flows, the mixing between the two phases is significant and the liquid flow can be either continuous or discontinuous. For horizontal pipes separation of the flow is greater due to the effect of the gravity but mixing can still occur depending on the velocity of the flow (Hewitt, 1982).

As a method, SPH is capable of simulating a bubble flow (Colagrossi and Landrini, 2003). However, any flow involving bubbles is a surface tension dominated problem. In order to predict correctly bubble separation and continue the simulation for a significant length of time, the use of an additional surface tension model is necessary (Rogers *et al.*, 2009). Such models have already been developed, including a model simulating the surface force

(Brackbill *et al.*, 1992), a colour function (Morris, 2000) and an acoustic Riemann solver (Leduc *et al.*, 2010).

Another process involving significant mixing of air and water flows is aeration. It is achieved by passing the air through the liquid via a diffuser or passing the liquid through air via fountains or cascades. Aeration is primarily used for treating industrial wastewater or increasing the amount of oxygen in the water (Bin, 1993). It is also used for producing smooth flows in faucets and aerated drinks. Creation of the latter is a very complicated process involving the infusion of the liquid with carbon dioxide and resulting in a bubble flow.

2.8. Concluding Remarks

This study will use the SPH formulation in order to simulate violent free-surface flows. Its meshless and Lagrangian nature make it ideal for nonlinear and violent cases as the free surface can be treated using only simple correction algorithms. The aim is to use the multi-phase formulation in order to simulate air-water flows in various fields such as coastal and nuclear engineering. Sufficient research has also been performed for multi-phase flows, allowing us to treat the density difference between the two fluids. Understanding the interactions between the different fluids is essential for these cases which include a diverse range of applications such as overturning waves and potentially explosive multi-phase pipe flow.

Successfully simulating these high-order phenomena requires an enormous number of particles, especially when using a multi-phase model. In order to simulate these particles the use of some form of parallel implementation of the program is necessary. This study will use GPUs as a tool for improving performance as they have proven to be as effective as massively parallel computer clusters, while being more cost-effective. Implementation on the GPU will be performed using the CUDA programming framework as it is currently the most developed and mature method as well as having been used extensively with SPH.

3. SPH Methodology

3.1. Introduction

The aim of the present chapter is to present the basic SPH methodology and models used in this study. The chapter will focus on the classical and the weakly compressible SPH formulation as well as the improvements and algorithms used in this study including the shifting algorithm. Special attention will be given to the model chosen for the simulation of interfacial flows (Colagrossi and Landrini, 2003).

3.2. General Principles

3.2.1. Interpolation

SPH is a Lagrangian method that does not require a mesh or a grid. The material is represented by local quantities at discrete Lagrangian locations, called particles. The simulation is advanced in time by computing the new position and properties of the particles using a time integration scheme. This can be applied to both fluids and solids.

The basis of the SPH formulation is an integral interpolation (Lucy, 1977, Gingold and Monaghan, 1977) of a function $A(\mathbf{r})$ defined over the domain Ω. The value of the function A is given at point $\mathbf{r}$ using a convolution product where the summation is obtained over the domain Ω:

$$A(\mathbf{r}) = \int_{\Omega} \delta_{\varepsilon}(\mathbf{r}-\mathbf{r}')A(\mathbf{r})\,\mathrm{d}\Omega \, . \tag{3.1}$$

In Equation (3.1) dΩ is a differential volume element and δ_ε is the Dirac function with an infinitesimally narrow region ε:

$$\delta_{\varepsilon}(x) = \lim_{\varepsilon \to 0} \begin{cases} 0 & x < -\varepsilon/2 \\ 1/\varepsilon & -\varepsilon/2 < x < \varepsilon/2 \\ 0 & x > \varepsilon/2 \end{cases} . \tag{3.2}$$

However, the Dirac delta function is infinitesimally narrow so it cannot be used in a computation. SPH approximates the delta function with the smoothing kernel W, another weighting function that approximates the delta function. The smoothing kernel depends on two quantities:

- The interpolation distance (distance between the particles)
- The smoothing length *h*, which is a characteristic length representing the extent of the kernel support and is linked to the distance between the particles. It is usually a constant but approaches using variable smoothing length have been developed (Gingold and Monaghan, 1982, Price and Monaghan, 2004). The radius of influence of the kernel is commonly twice the smoothing length to preserve the numerical stability of the method (Balsara, 1995). An example can be seen in Figure 3.1.

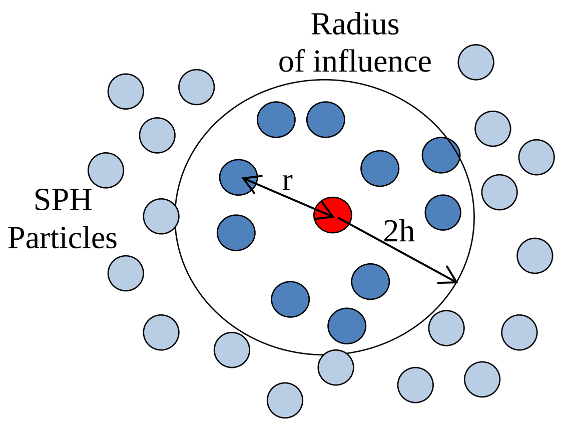


Figure 3.1: Smoothing kernel example

The SPH interpolation, now signified by the $\langle \bullet \rangle$ parentheses, becomes:

$$\langle A(\mathbf{r}) \rangle = \int_{\Omega} W(\mathbf{r}-\mathbf{r}') A(\mathbf{r}) \, \mathrm{d}\Omega \; . \tag{3.3}$$

It can be discretely approximated with a summation:

$$\langle A(\mathbf{r}_i) \rangle \approx \sum_{j=1}^{N} W(\mathbf{r}_i - \mathbf{r}_j, h) A(\mathbf{r}_j) \frac{m_j}{\rho_j} , \tag{3.4}$$

where *N* is the number of particles within the kernel limits, *j* denotes a neighbouring particle while *i* signifies the particle where the interpolation is centred. The volume element d*Ω* is essentially the volume of each particle and is replaced by the mass-density ratio. The new value of function *A* at the point $\mathbf{r}_i$ can be computed by the weighted summation of the adjacent points. The equation can be rewritten as follows:

$$\langle A_i \rangle \approx \sum_{j} W_{ij} A_j \frac{m_j}{\rho_j} , \tag{3.5}$$

where the subscript *i* or *j* denotes a value at the point occupied by the corresponding particle and:

$$W_{ij} = W(\mathbf{r}_i - \mathbf{r}_j, h) . \tag{3.6}$$

Based on Equation (3.4) we can find the formula directly for a gradient:

$$\langle \nabla A(\mathbf{r}_i) \rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j) \nabla_i W_{ij} \ , \tag{3.7}$$

where the term $\nabla_i W_{ij}$ denotes the gradient of the smoothing kernel with respect to the coordinates of particle *i*:

$$\nabla_i W_{ij} = \left(\frac{\partial}{\partial x_i} \mathbf{i} + \frac{\partial}{\partial y_i} \mathbf{j} + \frac{\partial}{\partial z_i} \mathbf{k} \right) W_{ij} \, , \tag{3.8}$$

where **i**, **j** and **k** are unit vectors in their coordinate directions. However, using Equation (3.7) for a function *A* will give a non-zero answer for the gradient, leading to significant errors (Monaghan, 1992). For that reason, we can use the following mathematical identity to compute the SPH gradient:

$$\rho \nabla A = \nabla(\rho A) - A \nabla \rho \, . \tag{3.9}$$

This identity leads to the following formulation for the SPH gradient:

$$\langle \nabla A_i \rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} \left(A_j - A_i \right) \nabla_i W_{ij} \, . \tag{3.10}$$

When modelling the forces between the particles neither the first, nor the second formula discussed in Equations (3.7) and (3.10) give an equal and opposite reaction (Monaghan, 1992). To create a gradient with these attributes the following identity can be used:

$$\frac{\nabla A}{\rho} = \nabla \left(\frac{A}{\rho} \right) - \frac{A}{\rho^2} \nabla \rho \, . \tag{3.11}$$

This identity leads to the following formulation for the SPH gradient:

$$\langle \nabla A_i \rangle = \sum_{j=1}^{N} m_j \left(\frac{A_j}{\rho_j^2} + \frac{A_i}{\rho_j^2} \right) \nabla_i W_{ij} \; . \tag{3.12}$$

The error of the integral interpolation can be found using a Taylor series expansion function *A(***r***′)* around **r** (Monaghan, 1992) the first two terms of which are displayed here:

$$A(\mathbf{r}') = A(\mathbf{r}) + (\mathbf{r}' - \mathbf{r}) \cdot \nabla A(\mathbf{r}) + O\left(|\mathbf{r}' - \mathbf{r}|^2 \right). \tag{3.13}$$

For the last term, the second-order truncation error, the order of the distance between positions **r** and **r***′* is generally similar to the order of the smoothing length *h*. If we use the SPH interpolation in Equation (3.13) we can derive:

$$\langle A(\mathbf{r}) \rangle = A(\mathbf{r}) \int_{\Omega} W(\mathbf{r} - \mathbf{r}', h) \, \mathrm{d}\Omega + \nabla A(\mathbf{r}) \int_{\Omega} (\mathbf{r}' - \mathbf{r}) W(\mathbf{r} - \mathbf{r}', h) \, \mathrm{d}\Omega + O\left(h^2\right). \tag{3.14}$$

The smoothing kernel is an even function with respect to **r** and therefore, the $(\mathbf{r}' - \mathbf{r}) W(\mathbf{r} - \mathbf{r}', h)$ is an odd function (Liu and Liu, 2003), making the second term equal to zero. Moreover, due to the kernel properties, which will be described in the next section, the first integral is equal to unity, leading us to:

$$\langle A(\mathbf{r}) \rangle = A(\mathbf{r}) + O\left(h^2\right). \tag{3.15}$$

We can then see that the basic SPH interpolation is at least second-order accurate. This result assumes that the integrals can be applied to the entire volume specified by the smoothing length. Near the boundary this does not apply, leading to reduced accuracy and possible numerical instabilities.

3.2.2. The Smoothing Kernel

The smoothing kernel is a function that needs to be chosen by the user. The kernel must tend to the delta function as the smoothing length approaches zero:

$$\lim_{h \to 0} W(\mathbf{r} - \mathbf{r}', h) = \delta(\mathbf{r} - \mathbf{r}'). \tag{3.16}$$

The kernel function should also be symmetric and equal to zero outside of its sphere of influence (which commonly of radius ±2*h*). It must be a differentiable function with a continuous derivative, with its integral in the interpolation region *Ω* being unity:

$$\int_{\Omega} W(\mathbf{r}-\mathbf{r}')\,\mathrm{d}\Omega = 1 . \tag{3.17}$$

There are several functions that fulfil these requirements. To ensure that the kernel integral is equal to unity, a normalisation factor α_D which differs depending on domain dimensions is used. The kernel is written using a non-dimensional variable *q* which is the particle distance over the smoothing length. The formulation, expressed along a single dimension, is:

$$q = \frac{r-r'}{h} . \tag{3.18}$$

Some common kernel choices include:

- Gaussian kernel, which is a very stable function even for high orders of derivatives (Liu and Liu, 2003), but is very expensive computationally as it is an infinite series:

$$W(r,h) = \alpha_D e^{-q^2} \quad \text{where } \alpha_D = \begin{cases} 1/\left(h\sqrt{\pi}\right) & \text{for 1D} \\ 1/\left(\pi h^2\right) & \text{for 2D} \\ 1/\left(\sqrt{\pi} h^3\right) & \text{for 3D} \end{cases} . \tag{3.19}$$

- the Schoenberg (1946) family of splines, which are piece-wise polynomials that approximate the Gaussian kernel instead of the Dirac function making them computationally cheap. There are several polynomials of different orders but the most commonly used is the cubic spline (Monaghan, 2005) although recently the quintic spline has also been used e.g. (Lind *et al.*, 2012b). The equation for the cubic spline is:

$$W(r,h) = \alpha_D \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \le q \le 1 \\ \frac{1}{4}(2-q)^3 & 1 \le \mathrm{q} \le 2 \end{cases} \quad \text{where } \alpha_D = \begin{cases} 2/3h & \text{for 1D} \\ 10/\left(7\pi h^2\right) & \text{for 2D} \\ 1/\left(\pi h^3\right) & \text{for 3D} \end{cases} . \tag{3.20}$$

- Quadratic kernel, which is a low-order polynomial and therefore computationally cheap. It has no extremum on its gradient, but has reduced accuracy because of its low order.

$$W(r,h)=\alpha_D\left[\frac{3}{16}q^2-\frac{3}{4}q+\frac{3}{4}\right] \text{ where } \alpha_D=\begin{cases}3/4h & \text{for 1D}\\ 2/(\pi h^2) & \text{for 2D}\\ 5/(4\pi h^3) & \text{for 3D}\end{cases}. \tag{3.21}$$

- Wendland (1995) quintic kernel, which has high order and can capture higher-order effects with improved accuracy, but is computationally expensive due to its high order. The equation is:

$$W(r,h)=\alpha_D\left(1-\frac{q}{2}\right)^4(2q+1) \text{ where } \alpha_D=\begin{matrix}3/(4h) & \text{for 1D}\\ 7/(4\pi h^2) & \text{for 2D}\\ 7/(8\pi h^3) & \text{for 3D}\end{matrix}. \tag{3.22}$$

For this project it is necessary to capture the more complex phenomena, such as overturning wave fronts so the use of a high-order kernel is necessary. The quintic kernels (spline and Wendland) offer a good middle ground between accuracy and required computational resources. Of these two 5th order kernels, the Wendland kernel will be used for this study since the quintic spline is a piece-wise polynomial and therefore less suitable for GPU programming as it will be explained in Chapter 4.

3.2.3. The Navier-Stokes Equations

In this study air-water flows are investigated. These flows are composed solely by fluids so they can be described by the Navier-Stokes (NS) equations. The equations expressing conservation of mass and momentum are given by:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t}=-\rho\nabla\cdot\mathbf{u}, \tag{3.23}$$

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}=-\frac{1}{\rho}\nabla p+\mathbf{g}+\frac{1}{\rho}(\nabla\cdot\rho\nu_o\nabla)\mathbf{u}, \tag{3.24}$$

$$\frac{\mathrm{d}e}{\mathrm{d}t}=-\frac{p}{\rho}\nabla\cdot\mathbf{u}, \tag{3.25}$$

where ρ is density, ν_0 is laminar viscosity, **u** is velocity, p is pressure, **g** signifies the external forces, e is energy and t is time. Body forces apart from gravity are not present, while in the energy equation only the changes in the thermal energy due to pressure differences are taken into account. Equations (3.23), (3.24) and (3.25) are the Navier-Stokes equations.

Using Equation (3.5) and substituting the function A with the density ρ it is possible to approximate the particle density (Monaghan, 1992):

$$\langle \rho_i \rangle \approx \sum_j m_j W_{ij} \,. \tag{3.26}$$

The SPH interpolation of the velocity gradient can be derived by using Equation (3.10) leading to the SPH version of the continuity equation:

$$\left\langle \frac{\mathrm{d}\rho}{\mathrm{d}t} \right\rangle = \sum_{j=1}^{N} m_j \left(\mathbf{u}_i - \mathbf{u}_j \right) \cdot \nabla_i W_{ij} \,. \tag{3.27}$$

Equation (3.27) is a difference formula that can correctly give the gradient of a constant field. It does not, however, ensure the incompressibility of the flow as the velocity divergence only approaches zero (Issa, 2005).

However, using Equation (3.10) for the momentum would not give an equal and opposite reaction between each set of particles and therefore would not conserve angular and linear momentum. The SPH gradient presented in Equation (3.12) will be used instead:

$$\left\langle \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} \right\rangle = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_j^2} \right) \nabla_i W_{ij} + \mathbf{g} \,. \tag{3.28}$$

Solving the energy equation is done in a similar way to the momentum to give:

$$\left\langle \frac{\mathrm{d}e}{\mathrm{d}t} \right\rangle = \sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_j^2} \right) \mathbf{u}_{ij} \cdot \nabla_i W_{ij} \,. \tag{3.29}$$

In the present study, the flows being simulated are adiabatic and no change of temperature occurs inside the system. Therefore, the energy equation is not needed for the simulation and will not be used in this study.

Particles can be moved by simply using:

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{u}_i \,. \tag{3.30}$$

Equations (3.27) and (3.28) are the classical SPH formulation used to solve the flow problems. To solve them however, one more condition needs to be satisfied: the mass of all particles within the system should remain constant:

$$\frac{\mathrm{d}m_i}{\mathrm{d}t} = 0 . \tag{3.31}$$

3.3. Additional models for free-surface flow

3.3.1. Pressure Computation

In order to close the SPH system presented earlier, an equation linking the pressure and the density is necessary. This is called an equation of state and can include either an algebraic or a differential formulation and differs depending on the phase of the material. For liquids and more specifically water there is a choice of 3 possible solutions:

a. Tait's equation of state adapted for fluid dynamics (Batchelor, 1967):

$$p_i = \frac{c_s^2 \rho_0}{\gamma}\left[\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right], \tag{3.32}$$

where ρ_0 is the reference density (for example in cases involving water in atmospheric conditions, we assume the reference conditions for temperature at 298K and density equal to 1000 kg/m^3) c_s is the speed of sound when the liquid density is equal to the reference density and γ is the polytropic index, depending on the material (7 for water). The subtraction of unity in the equation can remove the boundary effect for free-surface flows (Liu and Liu, 2003).

The large value of the polytropic index in fluids leads to a very sensitive formulation where small density variations can lead to large pressure fluctuations, regardless of the properties of the fluid. The compressibility of the equation depends on the speed of sound. Density variations are on the order of the Mach number squared (Monaghan, 1994):

$$\frac{\Delta\rho}{\rho_0} \sim \frac{u_{\max}^2}{c_s^2} = M^2 . \tag{3.33}$$

Hence, by keeping the Mach number in the region of 0-0.1, compressibility effects are on the order of 1% or less. As it will be shown on the next section however, the speed of sound is directly linked to the time step; using the actual value will result in a minuscule time step unsuitable for simulations. Hence, considering the balance between the size of the time step and the desirable incompressibility of the fluid, the speed of sound is selected relative to the maximum fluid velocity so that the density variations are kept below 1%.

b. A second option is a compressible equation of state, first introduced in SPH by (Morris *et al.*, 1997):

$$p_i = c_s^2(\rho_i - \rho_0). \tag{3.34}$$

This equation lacks the high order polynomial present in Tait's equation and is, as a result computationally faster and without the large pressure fluctuations. However, the fluid has a more compressible behaviour and the same restrictions apply to the speed of sound; increase of its value leads to a prohibitively small time step. Morris *et al.* (1997) propose a speed of sound value that allows for a density variation of 3%. The speed of sound should also be of the same order as the largest of the following factors:

$$c_s^2 \sim \frac{V^2}{\delta}, \frac{v_0 V}{L\delta}, \frac{FL_0}{\delta}, \tag{3.35}$$

where δ is the density variation, v_0 is the kinematic viscosity, V and L are velocity and length scales respectively and F is a body force per unit mass. The first factor corresponds to the one used in Tait's equation while the other two represent the viscous and body forces respectively.

c. The last option is to use a differential equation to ensure incompressibility, the pressure Poisson equation (PPE) (Lind *et al.*, 2012b):

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p \right)_i = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}_i^* \tag{3.36}$$

where $\mathbf{u}^*$ is an intermediate velocity obtained by advancing the solution over one time-step but ignoring the pressure. This is the equation used in incompressible SPH (ISPH) as it enforces the incompressibility of the fluid. Solving an SPH system with the Poisson equation requires different methodology such as the projection method (Cummins and Rudman, 1999) and is computationally very expensive. For free-surface flows, issues that lead to flow instabilities have been reported and a correction is needed for this equation to be used in such a case (Lind *et al.*, 2012b, Khayyer *et al.*, 2008).

From the three solutions to link the density and pressure, the first one (Tait's equation) will be selected for use in the present study, despite the large pressure variations. Ensuring the incompressibility of the water phase is essential in this study due to its interaction with the air which is a compressible phase. Therefore, Morris' equation, which allows for some compressibility in the liquid phase, cannot be used.

The Poisson equation would be suitable for these cases but the computational cost associated with discretising and solving an additional differential equation is prohibitive. There are also issues with the parallelisation of its discretised form due to the second order derivative; achieving an efficient algorithm with a massively parallel GPU system is something that, to the author's knowledge has not been investigated before.

Apart from water, in this study the modelling of air is necessary. For gases the basic equation of state is the ideal gas law:

$$p_i = \rho_i(\gamma - 1)e_i \tag{3.37}$$

where ρ is density, p is pressure, e is the thermal energy and γ is the polytropic index, depending on the material (1.4 for air). Simulating a system using the ideal gas law would then require solving the energy equation. There are several other possible formulations for gases depending on their temperature and state such as the Van der Waals equation of state:

$$\left(p_i + \frac{\alpha}{V_m^2}\right)(V_m - \beta) = RT_i \tag{3.38}$$

where V_m is molar volume, R is the universal gas constant and T is the temperature. The substance-specific constants α and β can be calculated from the material properties at the critical point. However, the gas used in the cases studied is air in atmospheric conditions with no minimal change in its temperature. Therefore, approximating it as an ideal gas would be sufficient. It has already been used as a basis for a multi-phase SPH model by Tartakovsky *et al.* (2009).

However, to avoid solving the energy equation, Colagrossi and Landrini (2003) have proposed the use of Tait's equation of state for the air phase. As it will be explained in the next section with more details, using different speed of sounds and a correction term it is possible to model air-water interaction without solving the energy equation.

3.3.2. Density filtering

Using Tait's equation of state, the pressure results have significant amounts of noise due to the reliance on the polytropic index which has large values for liquids. To solve this problem, density filtering is used where the density is re-initialised every few time steps (usually in the order of fifty). The correction happens using either a zeroth or a first order filter. The frequency of filtering needs to be carefully considered; applying it too frequently will eliminate the time evolution of the density.

A zeroth order filter, also called the Shepard filter, is a quick correction of the density field that is constantly applied every few time steps throughout the computation (Shepard, 1968):

$$\rho_i^{new} = \sum_j \rho_j \tilde{W}_{ij} \tag{3.39}$$

where the zeroth order correction is:

$$\tilde{W}_{ij} = \frac{W_{ij}}{\sum_j W_{ij} \frac{m_j}{\rho_j}} \tag{3.40}$$

The other option is to use a Moving Least Squares (MLS) scheme which is a first order correction that accurately reproduces the linear variation of the density field (Lancaster and Salkauskas, 1981).

Using the MLS filter requires inverting a 4x4 matrix for 3D (or a 3x3 matrix for 2D) for each fluid particle in the domain (Colagrossi and Landrini, 2003). This is a significant computational expense, especially since this procedure needs to be applied every few time steps. For that reason we will use the zeroth-order correction, the Shepard filter, which is a faster technique producing adequately accurate results.

3.3.3. Viscous Term

The momentum equation has so far been treated only in its inviscid form. To take the viscous term into account, an artificial term was proposed in order to model the effects (Monaghan and Gingold, 1983, Monaghan and Pongracic, 1985):

$$\Pi_{ij} = \begin{cases} -\dfrac{a\bar{c}_{ij}\mu_{ij} + \beta\mu_{ij}}{\bar{\rho}_{ij}} & (\mathbf{u}_i - \mathbf{u}_j)\cdot(\mathbf{r}_i - \mathbf{r}_j) < 0 \\ 0 & (\mathbf{u}_i - \mathbf{u}_j)\cdot(\mathbf{r}_i - \mathbf{r}_j) \geq 0 \end{cases} \tag{3.41}$$

where $\bar{\rho}_{ij}$ and $\bar{c}_{ij}$ are the mean value of the density and the speed of sound between particles *i* and *j* respectively and *α*, *β,* and μ_{ij} are parameters. The first two are empirical and depend on the case, while the latter one is given by the equation:

$$\mu_{ij} = \frac{h\mathbf{u}_{ij}\cdot\mathbf{r}_{ij}}{r_{ij}^2 + 0.01h^2} \tag{3.42}$$

In fluid dynamics the parameter *β* is usually not taken into account (Monaghan, 1992). This model has been used extensively in SPH for problems with low to medium Mach numbers (Monaghan, 2005). For high Mach numbers a different form of the viscosity was proposed based on the first dissipative term in shock simulations from Riemann solvers:

$$\Pi_{ij} = -\frac{Ku_{sig}(\mathbf{u}_{ij}\cdot\mathbf{r}_{ij})}{\bar{\rho}_{ij}|\mathbf{r}_{ij}|}, \quad \bar{\rho}_{ij} = \frac{\rho_i + \rho_j}{2} \tag{3.43}$$

where *K* is a parameter (with a value of approximately 0.5) and u_{sig} is a signal velocity given by the following equation:

$$u_{sig} = c_{si} + c_{sj} - \beta\mathbf{u}_{ij}\cdot\frac{\mathbf{r}_{\text{ij}}}{|\mathbf{r}_{\text{ij}}|} \tag{3.44}$$

This viscosity was tested for the dam break case studied in Chapter 5, but it was found that it significantly restricts the movement of particles in cases without high Mach numbers, simulating water as a more viscous fluid. In addition, an issue may arise with the equation of state where the speed of sound is selected to correspond to low velocities, so it will not be used in this study.

The artificial viscosity term is added to the momentum equation:

$$\left\langle \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \right\rangle = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \mathbf{g} \tag{3.45}$$

Another approach to modelling the viscous term was proposed by Morris *et al.* (1997) for laminar flows. They did not directly model the second order derivative due to the errors in the interpolation for low resolutions (Brookshaw, 1985) but used a combination of a standard SPH derivative with a finite difference approximation:

$$\left(v_o\nabla^2\mathbf{u}\right)_i = \sum_j m_j \frac{4v_0\left(\mathbf{u}_{ij}\cdot\mathbf{r}_{ij}\right)}{\left(\rho_i+\rho_j\right)\left|\mathbf{r}_{ij}\right|^2}\nabla_{\mathrm{i}}\mathrm{W}_{\mathrm{ij}} \tag{3.46}$$

The momentum equation then becomes:

$$\left\langle\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}\right\rangle = -\sum_{j=1}^{N} m_j\left(\frac{p_j}{\rho_j^2}+\frac{p_i}{\rho_j^2}\right)\nabla_i W_{ij} + \mathbf{g} + \sum_j m_j \frac{4v_0\left(\mathbf{u}_{ij}\cdot\mathbf{r}_{ij}\right)}{\left(\rho_i+\rho_j\right)\left|\mathbf{r}_{ij}\right|^2}\nabla_{\mathrm{i}}\mathrm{W}_{\mathrm{ij}} \tag{3.47}$$

Both the first artificial viscosity term and the laminar viscosity have been used in the cases presented in this study producing results with minimal differences. For the GPU code, since the accuracy of the viscous term is not the focus of this work and using the laminar viscosity is slightly more computationally expensive, the artificial viscosity term is preferred for the majority of the cases herein as will be shown in the next chapter.

3.3.4. Time integration

The time integration scheme that will be used should be at least a second-order accurate time scheme since a meshless method with moving interpolation points is being used. Four different time integration schemes were considered in this study and tested using a small FORTRAN code:

a. *Symplectic model:* The first one is a general symplectic scheme for an arbitrary function *A* (which in our case could be the velocity, the coordinates or the density). It is often known as the kick-drift-kick scheme where the kick is the velocity changing according to the force and drift is the coordinate changing with the initial velocity (Omidvar, 2010). It depends on two steps with a first evaluation being initially executed at half step for the density and the coordinates:

$$\begin{aligned}\rho_i^{n+1/2} &= \rho_i^n + \frac{\Delta t}{2}\frac{\mathrm{d}\rho}{\mathrm{d}t}_i^n \\ \mathbf{r}_i^{n+1/2} &= \mathbf{r}_i^n + \frac{\Delta t}{2}\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t}_i^n\end{aligned} \tag{3.48}$$

For the second step is the inter-particle forces are again obtained at the half step and the velocity and coordinates can be updated:

$$\begin{aligned}\mathbf{v}_{\mathrm{i}}^{n} &= \mathbf{v}_{i}^{n+1/2} + \frac{\Delta t}{2}\mathbf{F}_{i}^{n+1/2} \\ \mathbf{r}_{i}^{n+1} &= \mathbf{r}_{i}^{n+1/2} + \frac{\Delta t}{2}\mathbf{v}_{i}^{n+1}\end{aligned} \tag{3.49}$$

The values for the new density can then be computed using the velocity and the coordinate values:

$$\rho_{i}^{n+1} = \rho_{i}^{n+1/2} - \frac{\Delta t}{2}\frac{\mathrm{d}\rho}{\mathrm{d}t}_{i}^{n+1} \tag{3.50}$$

b. *Velocity Verlet model:* This is a formulation adapted from Molecular Dynamics (Verlet, 1967). Normally, the variables are calculated by the following equation:

$$A_{i}^{n+1} = A_{i}^{n-1} + 2\Delta t\frac{\mathrm{d}A_{i}^{n}}{\mathrm{d}t} \tag{3.51}$$

For the particle position Equation (3.51) changes slightly as the contribution of the inter-particle forces is taken into account:

$$\mathbf{r}_{i}^{n+1} = \mathbf{r}_{i}^{n} + \Delta t\mathbf{u}_{i}^{n} + 0.5\Delta t^{2}\frac{\mathrm{d}\mathbf{u}_{i}^{n}}{\mathrm{d}t} \tag{3.52}$$

The equations used in this scheme are therefore not coupled, so every few steps (on the order of 40 steps) the variables are computed using an Euler step in order to prevent the time integration diverging:

$$A_{i}^{n+1} = A_{i}^{n} + \Delta t\frac{\mathrm{d}A_{i}^{n}}{\mathrm{d}t} \tag{3.53}$$

c. *Leap-frog scheme:* The equations followed in this scheme are different depending on the variable advancing in time. In particular, the velocity is computed at the half-step using its gradient at the full step and the time steps of both the current and the previous step. The velocity at the current step can then be calculated as the mean value of the velocities at the current and previous half step while the new particle position is calculated using the velocities at the half step. The density, on the other hand, is the only variable in this scheme that can be updated without using the half step (Butcher, 2003).

d. *Runge-Kutta 4th order integrator (RK4):* This is a higher order method using multiple midpoints at the time step in order to eliminate lower-order error terms (Butcher, 1964).

In this study the time integration scheme used is the symplectic model. The RK4 model is more accurate, but the SPH equations need to be solved 4 times before proceeding to the time step, while the other methods examined only need at most two iterations. This offsets the increased time step size that can be achieved with this model. SPH has already a high computational cost as a method so adding the RK4 model will result in a very slow computation that will be unfeasible for large particle numbers.

The velocity Verlet model does not require the two iterations of the symplectic scheme, but the decoupling of the equations is a potential problem. The number of time steps the alternate formulation needs is not well defined and depends on the problem as well as the number of particles. The introduction of an arbitrary constant should be avoided whenever possible as the SPH scheme used in this study already includes a number of empirical constants, as it will be shown in the next section when detailing the multi-phase model.

The reasons for selecting the symplectic scheme over the leap-frog are purely computational. Creating a scheme that is based on two identical steps as opposed to a half step scheme solely for the momentum equation is simpler in a massively parallel system. In addition, the DualSPHysics code being used (detailed in Chapter 4) already includes a version of the symplectic model optimised for GPU architectures.

After the selection of the time integration scheme an appropriate time step must be selected. Due to the violent nature of the cases investigated using a constant time step is impossible; the conditions in the domain are constantly changing. A variable time step will then be used, based on the Courant-Friedrichs-Levy (CFL) condition.

There are three restrictions for selecting the time step. It can be dictated by the inter-particle forces (Monaghan, 1989) or the viscous conditions (Monaghan and Kos, 1999). The time step based on the forces is applied to ensure that particles do not move close to their neighbours:

$$\Delta t_f = \min_i \sqrt{\left(\frac{h}{|f_i|}\right)}, \tag{3.54}$$

where f_i is the internal or external forces applied to particle i. The restriction based on viscous forces is essentially based on the particle velocity and is given by the equation:

$$\Delta t_v = \min_i \frac{h}{c_s + \max_j \left| \frac{h\mathbf{u}_{ij} \cdot \mathbf{r}_{ij}}{\mathbf{r}_{ij}^2} \right|}. \tag{3.55}$$

A time step based on the artificial viscosity term in Equation (3.41) has also been proposed (Monaghan, 1989, Monaghan, 1992):

$$\Delta t_{cv} = \min_i \frac{h}{c_s + 0.6\left(\alpha c_s + \beta \max_j \mu_{ij}\right)}. \tag{3.56}$$

For the flows considered herein, the coefficient β in the last option is zero. The final time step is then selected as the smallest of the three and is multiplied by the CFL constant C_{CFL}:

$$\Delta t = C_{CFL} \min_i \left(\Delta t_f, \Delta t_v, \Delta t_{cv}\right). \tag{3.57}$$

For the simulations performed in the current study, it was found that the time step is primarily defined by the restrictions imposed by either the forces or the particle velocity, with the viscosity restriction having minimal to no effect on the simulation. The velocity restriction, due to the high speed of sound for the air phase, was the primary factor; however, on situations such as a wall impact, the large forces exerted on the particles further reduced the time step.

3.4. Model for the Simulation of interfacial Flows

For the simulations presented herein, we will be using the SPH formulation of Colagrossi and Landrini (2003) as this is a multi-phase SPH scheme has been proven effective with simulating violent air-water mixtures (Rogers *et al.*, 2009). Importantly, it is also the most straightforward to implement using CUDA on a GPU and hence investigate optimisation strategies and devise algorithms that will allow us to further reduce the computational time.

3.4.1. Equation of State

For an air-water mixture the Colagrossi and Landrini (2003) multi-phase model uses a simple procedure for predicting the interaction between water following on from the work of Nugent and Posch (2000), who propose the use of a modified version of Tait's equation of state (Batchelor, 1967) for incompressible and inviscid fluids:

$$p(\rho) = \frac{c_s^2 \rho_0}{\gamma}\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] + X - \bar{a}\rho^2 . \tag{3.58}$$

In this equation, X signifies the constant background pressure, a small pressure applied throughout the domain for numerical reasons, while the last term is a model of the cohesion forces between the particles of the same fluid with $\bar{a}$ being a cohesion coefficient. This term was first proposed by Nugent and Posch (2000) and the calculation of the coefficient is based on the properties of the different phases and the characteristic length of the domain:

$$\bar{a} = 1.5g\frac{\rho_{0,X}}{\rho_{0,Y}^2}L , \tag{3.59}$$

where $\rho_{0,X}$ and $\rho_{0,Y}$ are the initial densities of the two phases (water and air in this case). It is quite similar to the repulsive force term proposed by Monaghan (2011). Initially, it was proposed using the equivalent van der Waals parameter as the cohesion coefficient, but the accuracy was less than satisfactory so Equation (3.59) was used (Nugent and Posch, 2000). The characteristic length is an empirical coefficient that depends on the dimensions of the domain and the initial particle distance.

Colagrossi and Landrini (2003) used this term only for the fluid with the smaller density (which was usually a gas). The heavier fluid used Equation (3.32) for linking pressure and density. The purpose of this extra term is to prevent possible dispersion of the lighter fluid in the heavier especially with large density ratios and prevent the fragmentation of the interface by increasing the inter-particle forces in the phase it is applied.

As mentioned the equation of state heavily depends on the speed of sound. For the heavier fluid the speed of sound is selected in order to limit its compressibility. An initial value of the speed of sound can be found by Colagrossi and Landrini (2003):

$$c_{s,X} = \sqrt{\frac{p_0 \gamma_X}{\rho_{0,X}}},\ c_{s,Y} = \sqrt{\frac{p_0 \gamma_Y}{\rho_{0,Y}}}, \tag{3.60}$$

where p_0 is an initial pressure of the domain. Based on this equation, and substituting the actual speed of sound for a gas and a liquid (air and water for example) we can notice that the initial pressure values would not be on the same order of magnitude creating severe pressure discontinuities at the interface. In addition, the lighter fluid (the air) would be relatively less compressible than the heavier one (the water).

In order to remedy this situation the use of an unphysical value for the speed of sound is proposed. In order for the pressures at the interface to be similar the lighter fluid will be modelled by a higher sound speed in contrast to their actual physical values. In that sense, the speed of sound is now a coefficient determining the compressibility of the fluid that is being modelled rather than the actual physical quantity. The ratio of the different values can be found by Equation (3.61) assuming equal initial pressure:

$$\frac{c_{s,X}}{c_{s,Y}} = \sqrt{\frac{\rho_{0,Y} \gamma_X}{\rho_{0,X} \gamma_Y}}. \tag{3.61}$$

When determining a value for the speed of sound, the compressibility restriction in the value of the Mach number outlined by Equation (3.33) also needs to be taken into consideration. Suitable values for the speed of sound can then be found. However, since the stability of the method is directly linked to the speed of sound the higher speed for the lighter fluid means a decrease in the time step to prevent numerical instability. The time step is significantly decreased as the density ratio is increased, leading to large increases in the computational runtime.

3.4.2. SPH formulation

Colagrossi and Landrini (2003) show that the standard SPH formulation is not applicable in the multi-phase flows we are investigating, due to the large density discontinuity in the interface. The reason is the squared density term present in the SPH approximation of the momentum equation presented in Equation (3.28). With a large density ratio (the air-water ratio, for example is 1:1000), severe errors occur in the interpolation at the interface, leading to numerical instability.

To remedy this situation the following formulae have been used as proposed by Colagrossi and Landrini (2003):

Continuity:
$$\left\langle \frac{\mathrm{d}\rho_i}{\mathrm{d}t} \right\rangle = -\rho_i \sum_j \left[\left(\mathbf{u}_i - \mathbf{u}_j \right) \nabla_i W_{ij} \frac{m_j}{\rho_j} \right], \tag{3.62}$$

Momentum:
$$\left\langle \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \right\rangle = -\frac{1}{\rho_i} \sum_j \left[\left(p_i + p_j \right) \nabla_i W_{ij} \frac{m_j}{\rho_j} \right]. \tag{3.63}$$

The differences are the use of the density ratio in the continuity equation and the use of a different pressure gradient for the momentum equation (Bonet and Lok, 1999, Randles and Libersky, 1996). The new pressure gradient greatly diminishes the effect of the density discontinuities at the interface and is variationally consistent with the classical form (Bonet and Lok, 1999).

Following the work of Nugent and Posch (2000) on the equation of state, the use of an extra term in the momentum equation for the lighter phase is proposed. This term is also characterised by the cohesion coefficient and serves the same purpose: preventing the dispersion of the air particles in the water phase and the fragmentation of the interface. It is also only used for the lighter phase and the new momentum equation is (Colagrossi and Landrini, 2003):

$$\left\langle \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} \right\rangle = -\frac{1}{\rho_i} \sum_j \left(p_i + p_j \right) \nabla_i W_{ij} \frac{m_j}{\rho_j} - 2\bar{a}\rho_a^2 \frac{m_j}{\rho_j} \nabla_i W_{ij} . \tag{3.64}$$

Additional corrections have also been introduced to the velocity and the velocity divergence in order to improve the accuracy. For the velocity, the XSPH velocity correction, introduced by Monaghan (1989) is being used. This correction uses a mean velocity, originated from the velocities of the neighbouring particles. This equation is not used in the momentum equation, but rather in the equations for the position and the density:

$$\Delta \mathbf{u}_i = -\frac{\varepsilon}{2}\sum_j \frac{m_j}{\bar{\rho}_{ij}}\left(\mathbf{u}_i - \mathbf{u}_j\right)W_{ij} \tag{3.65}$$

Equation (3.65) is only used between particles of the same phase, ignoring the contribution of the other phase. This is also the case for the velocity divergence correction which is given by:

$$\nabla \cdot \mathbf{u}_i = \sum_j \frac{m_j}{\bar{\rho}_{ij}}\left(\mathbf{u}_i - \mathbf{u}_j\right)\cdot \nabla W_{ij} + \sum_j \frac{m_j}{\bar{\rho}_{ij}}\left(\Delta\mathbf{u}_i - \Delta\mathbf{u}_j\right)\cdot \nabla W_{ij} \tag{3.66}$$

The velocity divergence correction has minimal effect in free-surface flows (Monaghan, 1994) but is necessary when modelling bubble flows (Colagrossi and Landrini, 2003). A significant computing cost is involved with using this correction, as it must be applied for every particle. Rogers *et al.* (2009) found that the effect of this correction was insignificant both for the accuracy of the results and for the stability of gravity-driven simulations. For the XSPH velocity correction it was found that for the high resolutions used in this study, its impact was minimal and it has not been used for any of the simulations presented here.

A re-initialisation of the density field using a first-order interpolation scheme (moving least squares) was also proposed. The MLS scheme has already been mentioned in Section 3.3.2, but it requires the inversion of a 3x3 or a 4x4 matrix, greatly increasing the computational time. Therefore, the zeroth order correction for the density – the Shepard filter given in Equation (3.39) – will be used instead.

A different viscous term, modifying the artificial viscosity term of Monaghan and Pongracic (1985) is also proposed. The modified term is based on the work of Balsara (1995):

$$\mu_{ij} = h\frac{k_i + k_j}{2}\frac{\left(\mathbf{u}_i - \mathbf{u}_j\right)\cdot\left(\mathbf{r}_i - \mathbf{r}_j\right)}{\left|\mathbf{r}_i - \mathbf{r}_j\right|^2 + 0.01h^2} \tag{3.67}$$

$$k_i = \frac{\left|\nabla \cdot \mathbf{u}_i\right|}{\left|\nabla \cdot \mathbf{u}_i\right| + \left|\nabla \times \mathbf{u}_i\right| + 10^{-4}c_s / h} \tag{3.68}$$

This viscous term is a computationally intensive formulation to implement because of the divergence and curl terms which require the calculation of velocity gradients. Testing this term, it was found that the accuracy is not significantly increased since it is still based on the empirical coefficient $\bar{a}$, which needs to be changed for each different case. Therefore, the original implementation of the artificial viscosity will still be used.

3.5. Concluding Remarks

In this chapter the multi-phase model and the formulations that will be used in this study were detailed. Due to the relative simplicity and ease of implementation on GPUs, the Colagrossi and Landrini (2003) multi-phase model for the simulation of interfacial flows will be used with the modified equation of state and the NS SPH formulation. The artificial viscosity term by Monaghan and Pongracic (1985) and the zeroth-order density re-initialisation density filter (Shepard, 1968) will be used to ensure the numerical stability of the simulation.

In Chapter 5 the need for a particle shifting algorithm (Xu *et al.*, 2009, Lind *et al.,* 2012b, Skillen *et al.*, 2013) will also be tested for a multi-phase weakly compressible SPH formulation. Finally, a predictor-corrector scheme will be used for the time integration of the code with a variable time step restricted by the inter-particle forces and the viscous conditions of the simulation. The next chapter will present the implementation of this approach to the CUDA programming framework, as well as the steps taken for its optimisation.

4. SPH on GPUs

4.1. Introduction

This chapter presents the use of the CUDA programming framework for multi-phase SPH and the basic principles for an efficient and optimised parallel program on a GPU device. First, the structure and capabilities of the DualSPHysics code used in the present study are analysed and the modifications required in order to simulate a range of multi-phase cases are described. Different programming algorithms for the acceleration of the multi-phase simulations on graphics processing units (GPUs) are proposed. Their runtime results are examined in the end of this chapter for cases that enable evaluation of the GPU algorithms for multi-phase SPH: still water and a dam break case.

4.2. GPU Programming

4.2.1. Basic GPU architecture and capabilities

Graphics processing units (GPUs) are specialised hardware designed to process large quantities of data for displaying graphics on computer screens in real time. Primarily used for video games and image processing applications, their massively parallel architecture means they have emerged as a viable tool for scientific computing with architecture quite different to CPUs. Along with dedicated programming languages, such as CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language), general purpose graphics processing units (GPGPUs) have been extensively used for accelerating scientific computations.

Their primary use as a graphics rendering tool means that they are specialised for compute-intensive operations, ideal for meshless simulations which require extensive pairwise calculations. Compared to a CPU, a larger portion of the transistors on the chip is devoted to the Arithmetic Logic Unit (ALU) as opposed to data caching and flow control.

The difference in priorities can be seen in their parallel architecture. CPUs are characterised by multi-threaded cores, where each thread acts semi-independently. A GPU is based on single instruction multiple thread (SIMT) cores where the cores are much more dependent on an overarching command structure. This architecture makes them ideal for data-parallel computations where the same part of the program is executed simultaneously on many data elements. Such computations are characterised by an emphasis on arithmetic operations – a high arithmetic intensity – which comprise the majority of the simulation runtime.

For meshless applications, using the SIMT approach is advantageous for the computation as each particle requires an identical treatment. If, for example, it is assumed that a GPU has 400 cores (which is a number representative of most modern scientific GPUs) and each core has 32 threads (this is only assumed for simplification of the example, the number of threads per core depends on the data that is being processed) each computing the interactions of a single particle, then it is possible to compute data for 400×32=12800 particles simultaneously.

A memory structure is also present in the GPU; the global memory is shared among the multiprocessors, but each processor also has a shared memory where all its cores have access. The cores are all executing instructions in parallel, so synchronisation is a major issue and particular care must be taken when the cores are accessing or overwriting data.

On the GPU, there is a very high transfer rate between the memory and the processor, especially the transfer from the shared memory to the threads, which is at least an order of magnitude greater than the transfer rate between the global memory and the threads. The access time for data from the global memory however, is still smaller than accessing the main memory of the CPU. Transferring data to the CPU is a very slow process due to the low bandwidth compared to the memory bus (nVidia, 2012).

There are other memory subsets on the GPU: the constant memory and the texture memory. The first is a part of the global memory, but the values stored in that part, as the name implies, remain constant throughout the computation. It has a high transfer rate, but a small capacity (around 32KB). The texture memory is mainly used for graphics options and is able to store and transfer 2-D data arrays with a high speed. Using texture memory for storing numerical data, however, is difficult, due to the unique formatting these arrays require, which is based on spatial locality. Texture memory can also be used for filtering data.

4.2.2. GPU Programming Restrictions

A restriction of the GPUs is that the data simulated must fit on the GPU memory which currently, on the high-end cards, has an upper value of 6GB. The memory is then much smaller than the CPU RAM (as well as more expensive) and cannot be upgraded. This means that the data that can be used with a GPU-based program are limited and that GPUs are not as suitable for data-intensive applications.

This produces a programming challenge: in order to optimise the program the data needs to be stored in the GPU memory and for better optimisation exclusively in the shared memory. However, the latter is rather small so it must be used as efficiently as possible to avoid accessing the global memory. Data loading from the main CPU memory must also be minimised due to the lower transfer rate.

Moving data from the CPU to the GPU and vice versa is a low bandwidth operation that takes significant time. In most cases (Herault *et al.*, 2010), transferring the code exclusively to the GPU with the CPU only handling data control operations allows for lower runtimes, even if the tasks transferred are not inherently suited for parallel execution.

Another drawback of the GPU is that the computation of a logical function is slower than a numerical function. This is due to the selectively small amount of transistors devoted to data flow control in comparison to the ALU. Therefore, an effort to eliminate these functions and replace them with equivalent numerical statements should be made. This is especially important for a multi-phase simulation, where distinguishing between the different phases is vital due to the different terms and conditions employed for each phase.

Regarding the precision of the simulation, using either single or double precision is possible. Using double precision however, emphasises the drawbacks of the GPU: the added bytes required to store double precision variables limit the potential number of particles that can be simulated and the added flow control required significantly reduces the computational speed. In addition, due to the necessity for a high graphics output, GPU configuration gravitates towards the single precision option.

4.2.3. CUDA Programming Framework: Features and Limitations for Multi-phase SPH

As mentioned in Chapter 2 there are two existing frameworks for programming GPUs for scientific simulations. In this study, the tool being used in order to utilise the full potential of the GPUs will be the CUDA programming framework. It is based on the C/C++ programming language, with added features taking advantage of the parallel architecture. CUDA also handles any data transfers between the GPU and CPU.

The basic computing resource of a GPU can be considered the CUDA thread. These threads are organised into blocks which then form a computing grid, creating the basic internal hierarchy of a GPU card as shown in Figure 4.1:

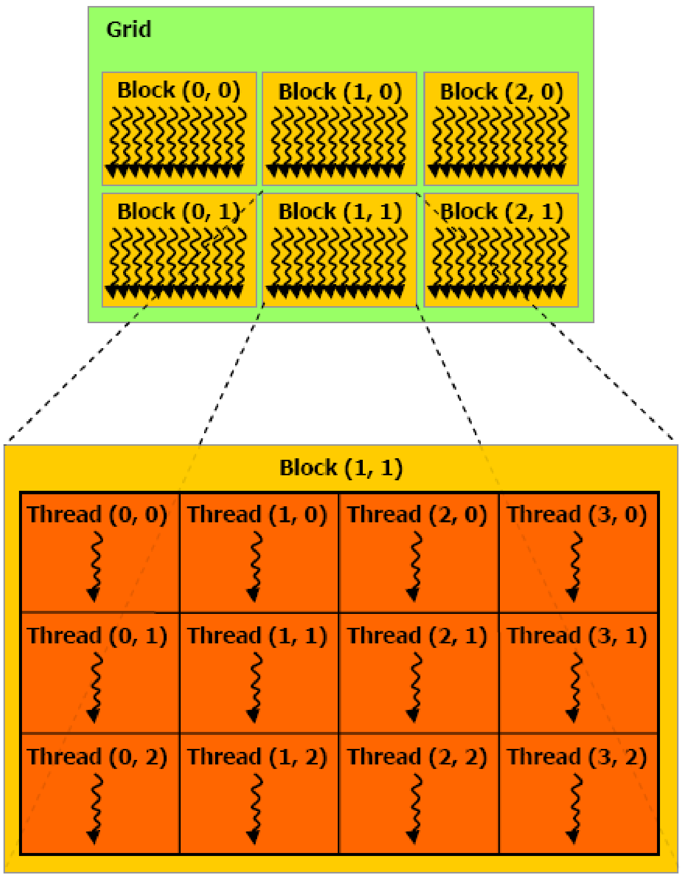


Figure 4.1: Grid of Thread Blocks (nVidia, 2012)

Threads in a block can cooperate when solving the same problem, where each block executes the associated commands individually, but multiple blocks can be executing the same part of the code. This leads to a hybrid system that uses coarse-grained parallelism in the block level and fine-grained parallelism in the thread level within each block. This approach ensures the scalability of the code as blocks can be freely assigned depending only on the simulation.

Considering the GPU hardware, each graphics card consists of several streaming multiprocessors (SMs). Threads within a block can only belong to the same processor; this creates an upper limit to the number of threads in each block. Within each block, they are organised by the multiprocessors in warps, each containing 32 threads. The number of threads in a block is, as a result, a multitude of 32, but the actual size of the block differs depending on the graphics card used and the case simulated.

Threads do not operate individually. When executing the code execution, the instructions are issued to each warp, not each thread individually. Threads within each warp then execute identical commands but for different elements in the domain (for a meshless method like SPH that would be different particles). This can be a problem with conditional statements, whose result can differ within the warp, leading to some threads having additional operations to perform, delaying the execution of the entire warp. Minimising the possibility of branching and limiting the amount of data necessary for the execution so that the shared memory is sufficient, are essential parameters for the optimisation of the code.

The number of blocks and warps within each multiprocessor depend on the kernels running and the memory amount each one is using. Among the warps, the multiprocessor has a specific portion of 32-bit registers for communication and data control, whose number is another limitation to the warps available to the processor. The amount of registers and memory available to each graphics unit are a function of the compute capability of the device. Better usage of the available resources includes then optimising the number of blocks and warps assigned to each kernel and minimising the latency between each warp command through synchronisation.

A different memory is associated with each level of thread hierarchy, shown in Figure 4.2. Each thread has a private local memory, but each thread in a block is also participating in a shared memory, which enables them to cooperate. These memories are on the same transistor chip as the thread, so transferring data has low latency and high bandwidth. The shared memory can be accessed simultaneously by the threads but care has to be taken to avoid data conflicts.

The data capacity for these two memories is limited, so most of the data is stored in the global GPU memory, which can be read by any grid but its access time for a thread is at least an order of magnitude longer than the shared memory. The constant and texture memories are also included in this subset but can only be written by the CPU and not by the GPU itself.

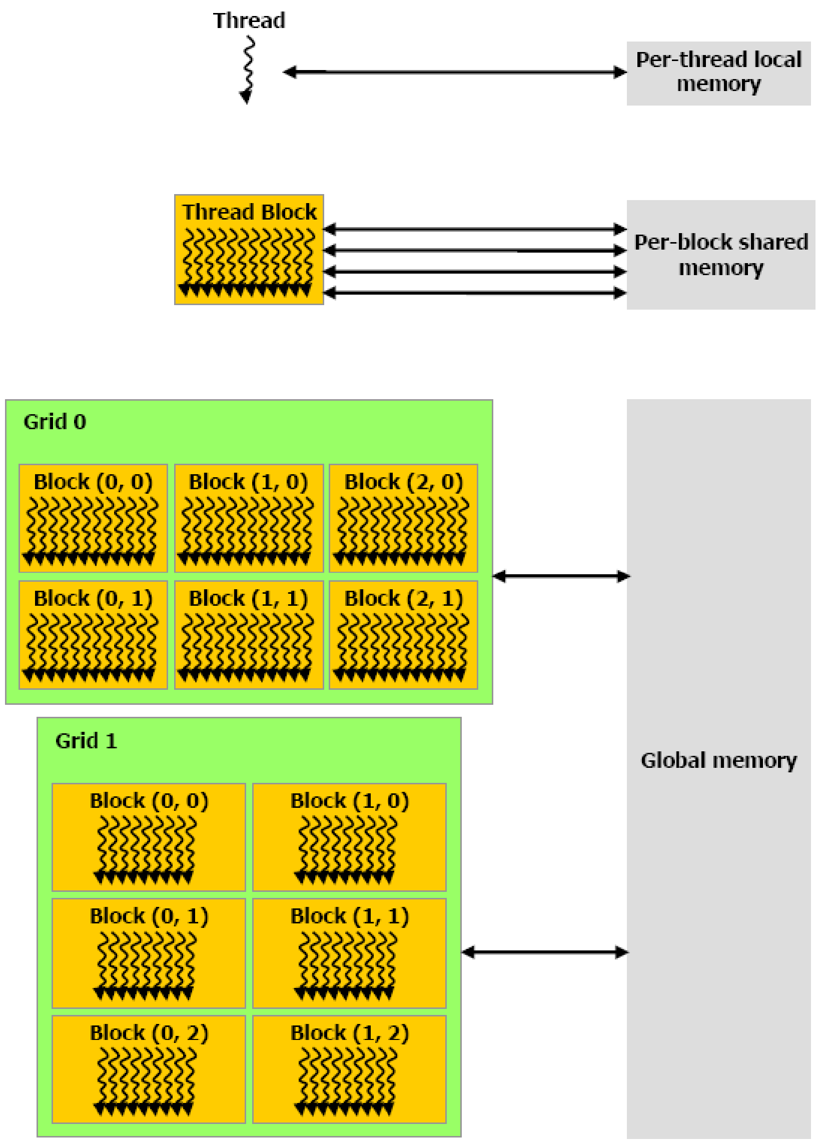


Figure 4.2: Memory Hierarchy (nVidia, 2012)

At a user level, application of the GPU resources depends on a hierarchy of user-defined functions and shared memories as well as synchronisation commands. The CUDA programming model separates execution calls to the GPU device from commands to the CPU host, which enables the creation of hybrid CPU-GPU programs. CUDA also distinguishes between the CPU and GPU memories and handles the transferring of the data.

These are accomplished by additional commands added to the existing C/C++ code. To enable the parallel aspect, CUDA provides each block and each thread within it with unique IDs. The program execution is scheduled by a series of C/C++ functions, called kernels. The kernels are executed in parallel multiple times by multiple CUDA blocks.

Kernel hierarchy is organised in three different levels. The host qualifier declares a kernel being executed and called by the CPU, while a global kernel is called by the CPU, but executed by the GPU, while the device kernel is called by and executed by the GPU. The latter cannot exist as a separate function and must be nested within a global kernel. It is possible for multiple device kernels to exist within a global one.

Based on this hierarchy, global kernels are mostly used for preparing variables for the computation as well as loading data from the global to the shared memory and vice versa, while the device kernels are used for the bulk of the numerical operations. Ideally, data used in the device kernels should be stored only in the local and shared memory to minimise accessing time. Device and global kernels should be designed while taking into consideration the inherent parallelism of the GPU as well as the possibility of asynchronous execution. On the contrary, host functions are more suitable for handling serial workloads.

Kernels can be launched asynchronously with the control returning to the host before the task has been completed. Concurrent kernel execution is possible as well. The possibility of asynchronous execution depends on the size of the kernel and the resources needed with larger kernels being more likely to operate individually. Control in asynchronous execution is achieved through streams, a string of orders executed sequentially.

4.3. The DualSPHysics Code

4.3.1. Introduction

The aim of this project is to create and implement a new multi-phase model for Smoothed Particle Hydrodynamics (SPH) using Graphical Processing Units (GPU). For this reason, the DualSPHysics code, developed by the University of Vigo in collaboration with the University of Manchester has been used (Gomez-Gesteira *et al.*, 2012). The version used in the present study is 2.01.

The code consists of two parts: a pure CPU approach written in C++ and a hybrid CPU-GPU approach utilising the CUDA framework. This has been selected in order to offer versatility, enabling a CPU-based SPH simulation even when a CUDA-enabled card is not available. Comparison of the CPU and GPU results are also straightforward. The DualSPHysics code has been developed from the SPHysics FORTRAN code, but not all the models and capabilities of the parent code have been transferred thus far.

The CPU and CPU-GPU codes are not completely separate with some parts of the code being shared between the two versions. These parts are coded in C++ and are executed on the CPU and involve loading the initial configuration of the simulation as well as saving the results to an appropriate file designated by the user.

4.3.2. Code Structure

The DualSPHysics GPU SPH simulation consists of three main steps: (i) creation of the particle neighbour list, (ii) particle interaction and (iii) the variable update at the end of the time step. This whole system is constantly repeated until the end of the simulation. To ensure the maximum efficiency for the GPU and to create a more intuitive program each thread computes data only for a single particle at a time.

The structure of the code is similar for both approaches and can be seen in Figure 4.3:

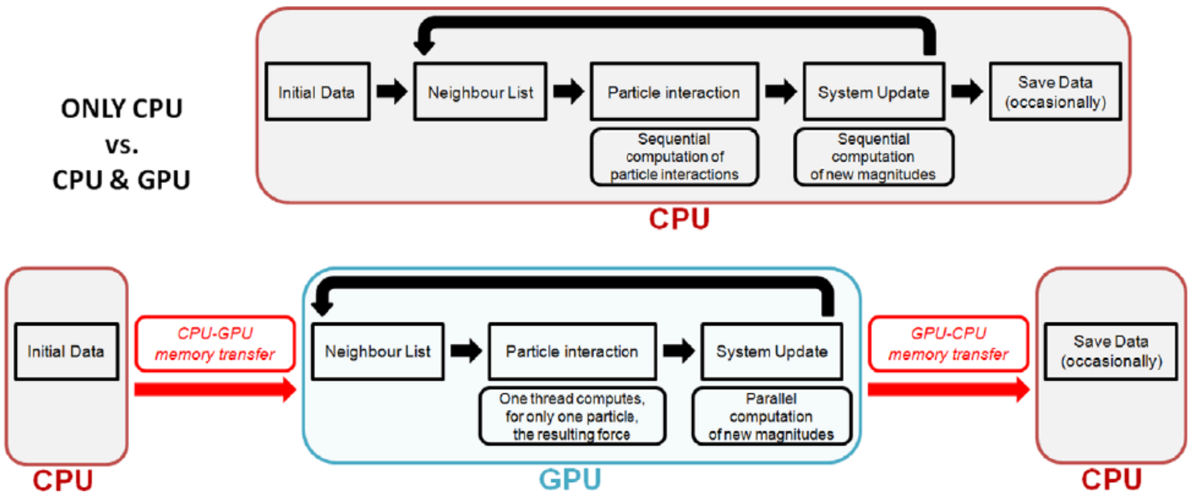


Figure 4.3: Main structure of the DualSPHysics code (Crespo *et al.*, 2011a)

The CPU implementation of the DualSPHysics code has been optimised based on the experience acquired from the SPHysics FORTRAN code. However, the main focus of this study is the hybrid CPU-GPU approach due to the lower runtimes and the capability for simulating a larger number of particles. As we can see from Figure 4.3 particle pre-processing and storing is handled by the CPU, while all the other code is being run on the device as per the guidelines given earlier to minimise CPU-GPU interaction.

i) Neighbour list construction

In each time step, in order to create the neighbour lists and perform the force computations, access to a large quantity of data requires a large use of GPU memory. The access order of this data, as well as the way they are stored in the GPU shared memory is important for the reduction of the computational runtime. As shown by Hérault *et al.* (2010) and Crespo *et al.* (2010) storing data required by the same thread in neighbouring memory addresses has a positive impact on performance.

Finding the neighbouring particles in SPH requires computing pair-wise distances which can be a very expensive computational procedure and is unfeasible for large domains due to the large number of particles involved. The DualSPHysics code uses the cell-linked list approach (CLL) (Dominguez *et al.*, 2010) where the domain is divided into cells whose dimension is the kernel radius such that particles are stored according to the cell to which they belong (see Figure 4.4). This approach ensures that any particle in a cell needs to look for its neighbouring particles only in the adjacent cells and not the whole domain. The downside is the greater memory requirement (Dominguez *et al.*, 2010).

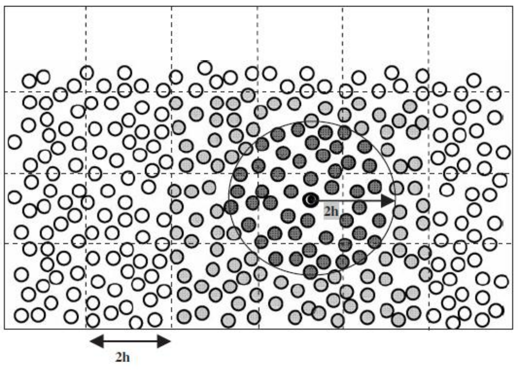


Figure 4.4: Example of the Cell-linked List (Dominguez *et al.*, 2010)

Due to the large deformations present in many SPH simulations the particles' initial order, as well as the cell in which they belong will change during the computation. For maximum speed up on a GPU, reordering of the particle lists according to their new position enables the access order to the data to be optimised every time step (the reordering uses the THRUST algorithm which is included in the intrinsic CUDA libraries). The main hydrodynamic variables of the particles are also re-ordered using the same approach.

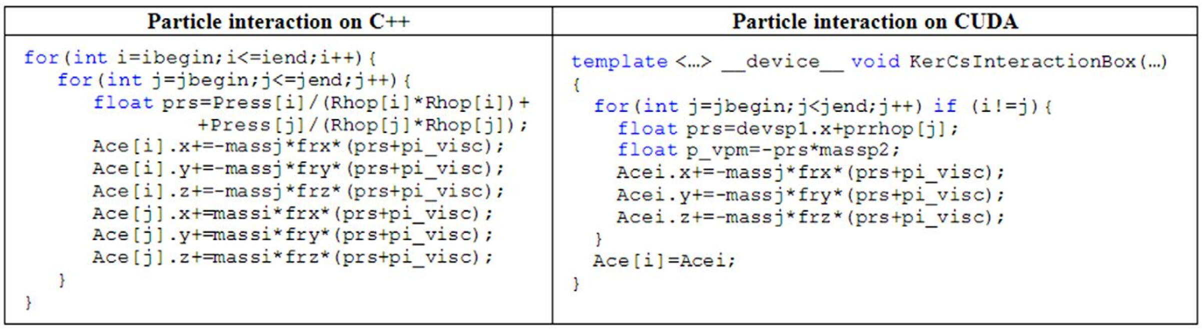
ii) Force computation

The next step in the SPH simulation is to solve the momentum and continuity equations calculating the forces among the particles as well as the new pressure. The equation of state used is Tait's weakly compressible equation of state (Batchelor, 1967). In general, the original DualSPHysics uses the classical SPH equations (Equations (3.27) and (3.28) with the addition of the Shepard filter (Shepard, 1968), presented in Equation (3.39) and the XSPH velocity correction by Monaghan (1989).

A choice can be made between two different models for both the viscous term and the kernel function. For the former, the artificial viscosity term (Monaghan and Gingold, 1983, Monaghan and Pongracic, 1985) and the laminar SPH viscosity (Morris *et al.*, 1997) are available. The latter includes the SPS turbulence model introduced by Dalrymple and Rogers (2006). It is not used in the computations outlined here as turbulence is beyond the scope of this study. For the kernel, the cubic spline function (Schoenberg, 1946) and the Wendland quintic kernel (Wendland, 1995), given by equations (3.20) and (3.22) respectively, are both available. The latter will be used in this study for reasons explained in Section 3.2.2.

To create boundaries, DualSPHysics uses the fictitious boundaries approach (Crespo *et al.*, 2007) mentioned in Section 2.3.2, creating new particles with the hydrodynamic attributes of the fluid. This method takes advantage of the parallel nature of the GPU, as each boundary particle can be treated by a single thread. Their treatment is identical to the fluid particle, excluding the momentum equation, so their inclusion does not greatly increase the complexity of the system.

An important difference between the CPU and the GPU SPH solver is the application of the kernel symmetry. This kernel attribute extends symmetry to the pair-wise particle forces so that when calculating the interaction of a particle with his neighbour, the force exerted by each article on its neighbour has the same magnitude but opposite direction. For the CPU, the kernel symmetry can be used to update two particles in a single iteration of the solver. This approach cannot be used on the GPU because of the simultaneous access to memory from parallel threads. We can see the difference in the code in Figure 4.5.



Particle interaction on C++

```
for(int i=ibegin;i<=iend;i++){
   for(int j=jbegin;j<=jend;j++){
      float prs=Press[i]/(Rhop[i]*Rhop[i])+
               +Press[j]/(Rhop[j]*Rhop[j]);
      Ace[i].x+=-massj*frx*(prs+pi_visc);
      Ace[i].y+=-massj*fry*(prs+pi_visc);
      Ace[i].z+=-massj*frz*(prs+pi_visc);
      Ace[j].x+=massi*frx*(prs+pi_visc);
      Ace[j].y+=massi*fry*(prs+pi_visc);
      Ace[j].z+=massi*frz*(prs+pi_visc);
   }
}
```

Particle interaction on CUDA

```
template <…> __device__ void KerCsInteractionBox(…)
{
  for(int j=jbegin;j<jend;j++) if (i!=j){
    float prs=devsp1.x+prrhop[j];
    float p_vpm=-prs*massp2;
    Acei.x+=-massj*frx*(prs+pi_visc);
    Acei.y+=-massj*fry*(prs+pi_visc);
    Acei.z+=-massj*frz*(prs+pi_visc);
  }
  Ace[i]=Acei;
}
```

Figure 4.5: Pseudocode of the particle interaction procedure implemented on CPU and GPU (Crespo *et al.*, 2011a)

iii) System update

The last part of the code deals with updating the hydrodynamic variables of the system as the simulation advances in time. A variable time step is used with the particle forces and the particle velocity restricting the maximum time step value allowed. Two second-order time schemes can be used: the velocity Verlet scheme (Verlet, 1967) and the symplectic scheme (Dullweber *et al.*, 1997). Used in this study is the second scheme, which was detailed in Section 3.3.4.

During the update process, the particle data are occasionally saved in the CPU. The format of the files for saving the particle data affects the computation runtime and files of smaller size, such as binary files correspond to a lower runtime.

Regarding the file structure of the code, both the CPU and the GPU code have been separated to several files and subroutines, which are interconnected and called by the main routine when and if they are needed. This modular approach allows for an easy understanding of the code as well as simplifying the issue of adding additional models or features to the code.

4.3.3. Pre-and Post-Processing for DualSPHysics

Before the start of the computation however, an input needs to be created. The DualSPHysics code uses files provided by a pre-processing program, called GenCase. The GenCase program uses an XML file to define the initial geometry and configuration of the case as well as provide the DualSPHysics code the initial parameters needed for the simulation. GenCase also allows for the simulation of floating bodies as well as particle motion can be simulated.

To allow the visualising of the simulation results as well as facilitating their analysis, several additional programs are supplied with the DualSPHysics code for visualisation (BoundaryVTK, PartVTK) and data analysis (MeasureTool, IsoSurface). These tools are also used in the present study.

The DualSPHysics code has already been used to simulate a number of free-surface flows, giving good agreement with validation data. This includes validation using the SPHERIC benchmark test case 2 (Kleefsman *et al.*, 2005), a 3-D dam break over an obstacle, which tracks the evolution of the free surface as well as the forces exerted on the obstacle. The experimental results were compared to several simulations with different resolutions and the results were satisfactory (Crespo *et al.*, 2011b, Crespo *et al.*, 2011a). The DualSPHysics code has also been used for simulating sediment suspension (Fourtakas *et al.*, 2013a) as well as computing the forces exerted on a beach by large waves (Barreiro *et al.*, 2013). Compared to a single-core CPU simulation, DualSPHysics can give speed-ups of up to 2 orders of magnitude (Crespo *et al.*, 2011a).

4.4. Modification of DualSPHysics for the Multi-Phase Model

The DualSPHysics code described so far is only able to simulate single-phase cases, significantly restricting its usability and the number of applications that is possible to simulate. By applying a multi-phase model, the program becomes more robust and versatile and the benefits, such as the decrease of the computational runtime, gained by the use of the GPU can be applied to multi-phase problems.

The application of DualSPHysics to multi-phase problems greatly benefits from the additional processing power and increase in parallelism offered by the GPU as simulating multiple phases is usually associated with a larger number of particles and a larger domain. In addition, interaction among the phases leads to high-order flow phenomena, which need a high particle resolution to be simulated accurately.

In order to implement the multi-phase model, several changes to the single-phase program need to be implemented. These changes do not alter the main program structure, preserving the limited communication between CPU and GPU and the improvements gained over the CPU-only implementation. Modifications focus instead on the inclusion of the new terms advocated by the Colagrossi-Landrini model for the second phase and the optimisation of the code as deemed necessary by these changes. The required changes have been implemented in both the CPU and the GPU code.

The time-stepping algorithms present in DualSPHysics (Verlet and symplectic) and the part of the code that updates the particle values for the new time step have not been significantly altered. The only change in this part is in the calculation of the new time step value. The time step restrictions outlined in Section 3.3.4 will be calculated twice and updated for both phases with the smallest time step being selected for the new iteration.

4.4.1. Updating Neighbour and Cell lists

Before changing the main DualSPHysics code, the pre-processing tool GenCase has been modified in collaboration with the University of Vigo in order to support input for multiple fluids. The modified version allows us to define the initial geometry of both fluids as well as some initial reference attributes such as the speed of sound and the initial density. The different phases are created as separate volumes within the computational domain. The new modified pre-processing tool can be used with both the multi-phase and the initial version of the code.

The introduction of the reference attributes necessitated changes in their treatment compared to the single phase as they were considered as constants in the initial code. However, as the neighbour and cell lists are updated, these values would need to be reordered using the THRUST algorithm every time step, which is costly computationally. Instead, these values were stored in the constant memory, which offers a lower latency.

Despite being defined as different volumes in the pre-processing tool, particle data in the main code is stored in a single array for each hydrodynamic variable. The program is however already able to distinguish between boundary and the fluid particles using the data provided by GenCase and the code can create separate cell and neighbour lists depending on the nature of the particle. This difference also allows the use of different CUDA kernels for each list. This approach eliminates the use of conditional statements, which introduce branching in the code and potentially limit the speed of the computation.

The same principle can be applied to a second fluid phase. The number of particles belonging to either water or air can be extracted by the pre-processing tool; particles do not change phases during the computation. It is then possible to create separate cell and neighbour lists for each phase enabling the use of separate CUDA kernels.

There are however advantages to treating the fluid particles as a single list, such as a reduced complexity of the code, launching a smaller number of CUDA kernels and handling a smaller amount of data. In addition, despite the extra terms, treatment of the different phases is fundamentally similar, unlike the boundaries. In that case, the creation of the cells in the multi-phase code would follow an identical approach to the single-phase code: they are created in order to optimise data access for the CUDA kernels. As a result a cell can potentially contain both air and water particles.

4.4.2. Modifying Force Computation for Multi-Phase SPH

The main change in the force computation part of the DualSPHysics code is updating the equation of state to the forms specified by the multi-phase model and mentioned in Section 3.4.1. The Navier-Stokes equations are also updated to the new forms, outlined in Section 3.4.2 but there is an additional numerical consideration:

Solving the momentum and continuity equations requires a summation of all the contributions of the neighbouring particles. If the values of these contributions have large differences it is possible, when using the single precision, for the final value to be incorrect due to the round-off error by the finite memory associated to each variable. This error is proportional to the number of variables being summed so it increases as the particle resolution also increases.

When including the air phase, the difference between the values for each variable can be several orders of magnitude, especially in some cases such as the density. This leads to the smaller values being ignored, an error that propagates through the computation. Due to the equation of state, which for water includes raising a value to the seventh power, small differences in density can account for large variations in the pressure field.

To ensure then that the summation is correct, the Kahan summation algorithm has been introduced (Kahan, 1965) as suggested by Longshaw (2013). This algorithm uses a separate variable to record small error (a running compensation) which is then added to the final summation value. With this algorithm the error is independent of the number of variables being summed and only depends on the floating-point precision (single, in this study). The algorithm is displayed in the following table:

```
function KahanSum(input)
    sum = 0.0                            // Summation value and output variable
    comp = 0.0                           // Running compensation
    for i = 1 to number_of_input_values
        x = input[i] - comp              // Correct input value using the compensation
        y = sum + x                      // Calculate new summation
        comp = (y – sum) – x             // Calculate new compensation (ideally
        sum = y                          equals 0)
        return sum
end
```

Table 2: Pseudocode of the Kahan summation algorithm

Implementing the summation algorithm for the cases presented herein led to a decrease in the summation error by 1-2% depending on the number of particles and the case examined. Results did not show any significant improvement, indicating that the summation error is too small to significantly alter the results of the computation.

Apart from the issue of solving the Navier-Stokes equations, a different issue arises when considering the treatment of the particles within the kernels that perform the force computation operations. The new equations require separate treatment for each phase and a means to facilitate that must be found. There exist two possible solutions for this issue:

i) The initial code differentiates among the particles in each cell by using the thread number. The index number of each thread within a block is unique and since each block operates independently, accessing different sets of data and each thread computes only the values for a single particle; the resulting particle number is unique.

   Applying this approach for multiple fluids is not possible for a single list, as the thread number constantly changes. In this case, DualSPHysics also assigns to each particle a unique identification number (referred to as IDs herein). These IDs are stored in a separate array which remains constant even when the particles are reordered. Using the particle IDs it is possible to differentiate between the phases; however, additional conditional statements are required.

ii) Using the thread number for the different cases is possible in the event of multiple cell and neighbour lists. The CUDA kernels launched in this case, however, must have as input particles of a single phase only, otherwise conditional statements are required and any advantage of using multiple lists is negated. Restricting the kernel input can potentially lead to a sub-optimal arrangement of the particle data in the GPU memory, increasing the access and loading times and reducing the CUDA kernel occupancy.

The solutions proposed each have unique advantages and disadvantages. Implementing either of these approaches require significant alterations in the code. In addition these alterations will have to be consistent with the neighbour lists created earlier. Investigation of the methods described led to the creation of four new algorithms spanning the entire structure of the code. These algorithms are presented in the next section.

4.5. New algorithms for optimising the multi-phase treatment

At the beginning of this study no answer to the optimal treatment of a multi-phase case was available. An investigation into this matter was deemed necessary as lower computational runtimes is the biggest advantage offered by the use of a GPU, leading to higher resolution cases. For that reason four different algorithms were investigated to answer the issues described. A more detailed discussion of these algorithms follows.

4.5.1. Algorithm 1: Using Conditional *if*-statements

The DualSPHysics code includes every fluid particle in a single list even when a second phase is being introduced. As mentioned, with the default scheme for distinguishing among particles this approach will lead to significant errors as different terms are needed for each phase. One solution is the use of conditional *if* statements.

With the data of the particle generation it is possible to create a simple, yet effective filter which will separate the phases based on the particle IDs (these remain constant throughout the computation). This filter mainly needs to be applied during the calculation of the equation of state and the momentum equation, as their forms are different depending on the phase of the primary particle.

This algorithm has the advantage of being relatively simple; retaining the cell and neighbour lists of the original program without increasing the number of times the THRUST algorithm is called. The changes in the structure of the original code are also limited, retaining the minimal interaction between the GPU and the CPU.

The main problem of this approach is an issue inherent with GPU programming, using logical and conditional statements is time consuming. The conditional statement needs to be used every time the equation of state is computed or any interaction among particles is being computed, its total number of executions in each time step being over twice the number of particles in the system, adding a considerable overhead time.

Another issue is the complexity of the main CUDA kernel. In DualSPHysics, several main functions of the code (neighbour search, calculation of the Navier-Stokes equations) are performed in a single kernel. This CUDA kernel now includes the ID filter for each phase and the multi-phase model increasing the memory and the resources (threads, registers) the GPU needs to provide for its computation.

Such a large kernel means that a concurrent launch with other CUDA kernels is almost impossible due to the amount of blocks needed for its execution. It is also probable the large amount of data, which depends on the number of particles, cannot be loaded into the shared memory and must be directly accessed form the GPU global memory with a lower bandwidth. This kernel may very well act as a bottleneck for the whole system.

4.5.2. Algorithm 2: Conditional Binary Multipliers

An alternative approach makes use of conditional operators instead of *if* statements. A conditional operator in CUDA/C++ evaluates a binary expression returning a value depending on its result. This value can then be stored in a new variable. Similar to the first algorithm, we then use the particle IDs to change its outcome and distinguish between the different phases via the operator.

Since the conditional operator is a multiplication operation, this is a faster alternative to *if* statements when selecting a single value for a given phase and is the preferred approach in all cases for applying a single-value that is uniform for an individual phase. Importantly for GPU, it limits the branching between the threads.

As an example, Table 3 shows the use of a binary multiplier for the computing the pressure in the equation of state:

Original code:	Modified code:
`If (ParticleID<CT.WaterLimitID){`	`PhaseSwitch=(ParticleID<CT.WaterLimitID?0:1);`
`   Pressure=B*(powf(Rhop/CT.RhopWater,`	
`   CT.GammaWater)-1.0f)+X`	`Rhop0=CT.RhopWater*(1-PhaseSwitch)+CT.RhopAir*PhaseSwitch;`
`Else`	`Gamma=CT.GammaWater*(1-PhaseSwitch)+CT.GammaAir*PhaseSwitch;`
`   Pressure=B*(powf(Rhop/CT.RhopAir,`	
`   CT.GammaAir)-1.0f)-CL*Rhop^2+X`	`Pressure=B*(powf(Rhop/Rhop0, Gamma)-1.0f)-CL*Rhop^2+X`
`Endif`	

Table 3: Pseudocode of the equation of state at Algorithms 1 and 2 where CT denotes a fluid constant

The values changing through the particle IDs in this statement can be simple integer numbers so the additional amount of data being processed is minimal. It is similar to the first approach, meaning that the changes required to the code are small and the interaction between the CPU and GPU are still minimal.

The most important issue compared to the first approach is the increased number of computations. All the interactions between the particles use this approach, including the momentum equation and the equation of state. In the code, this translates to six additional operations per particle and nine additional computations for each of its neighbours for each iteration. Considering that the aim is the simulation of millions of particles, this is a significant load even if arithmetic operations on the GPU are fast.

This approach investigates whether removing branching and simplifying the conditional statements will counterbalance the addition of more operations. Note that, because of the small changes from the first approach, the issue of a complex, resource-intensive kernel remains.

4.5.3. Algorithm 3: Separate Particle Lists for each Phase

The resources inside each GPU (threads, registers and blocks) are finite. Using a single CUDA kernel that performs all the operations involved in a single iteration of an SPH computation is a potentially inefficient and prohibitive use of the GPU resources. Therefore, using a CUDA kernel that loads particle data to the GPU device, determines the neighbour lists and performs all fluid particle interactions inside the domain has the potential to impact the simulation severely.

As mentioned when discussing the first algorithm, it is possible for a bottleneck to be formed when the GPU needs to dedicate all its resources to computing this kernel. The computational time is then defined by the speed in which this kernel is being executed, regardless of any other optimisations within the code. This situation becomes progressively worse as the number of particles increases. This is a result of a unified cell and neighbour list.

Similarly for data transfer between GPU and CPU, when accessing the device kernel transferring more data means that the GPU memory needs to be accessed multiple times during the iteration. Furthermore, if the GPU shared memory is insufficient for the data, the program will need to access data in the global memory, whose access time is much slower (Herault *et al.*, 2010).

To reduce the data processing required by the CUDA kernel, the particle lists for the different phases can be separated (similar to sorting the fluid particles from the boundary particles in the original DualSPHysics code). Splitting the cell list of the fluid particles in smaller, distinct ones for each phase and creating a different neighbour list for each particle as it interacts with different phases is an option that can potentially remove that bottleneck.

Figure 4.6 shows the differences between this approach and the previous two algorithms. Similar to the original DualSPHysics code, the first algorithms have one list for all fluid particles, while the new multi-phase scheme has separate lists for each phase.

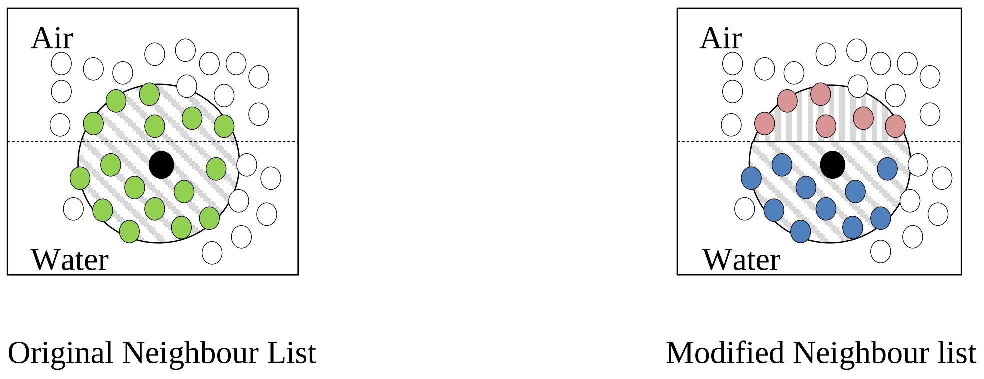


Figure 4.6: Neighbour Lists for single and multi-phase DualSPHysics GPU code (green shows the previous neighbour list, while red and blue note the new lists for air and water phase respectively)

The cell lists are created at the beginning of each iteration on the GPU after the particles are reordered following the system update from the previous time step. The different cell lists allows the use of a separate global CUDA kernel for each phase which computes the particle interactions. Within each of these global CUDA kernels, there are additional device kernels (only called and executed by the GPU) which compute the interactions with the neighbour lists of the different phases. This is displayed in Figure 4.7.

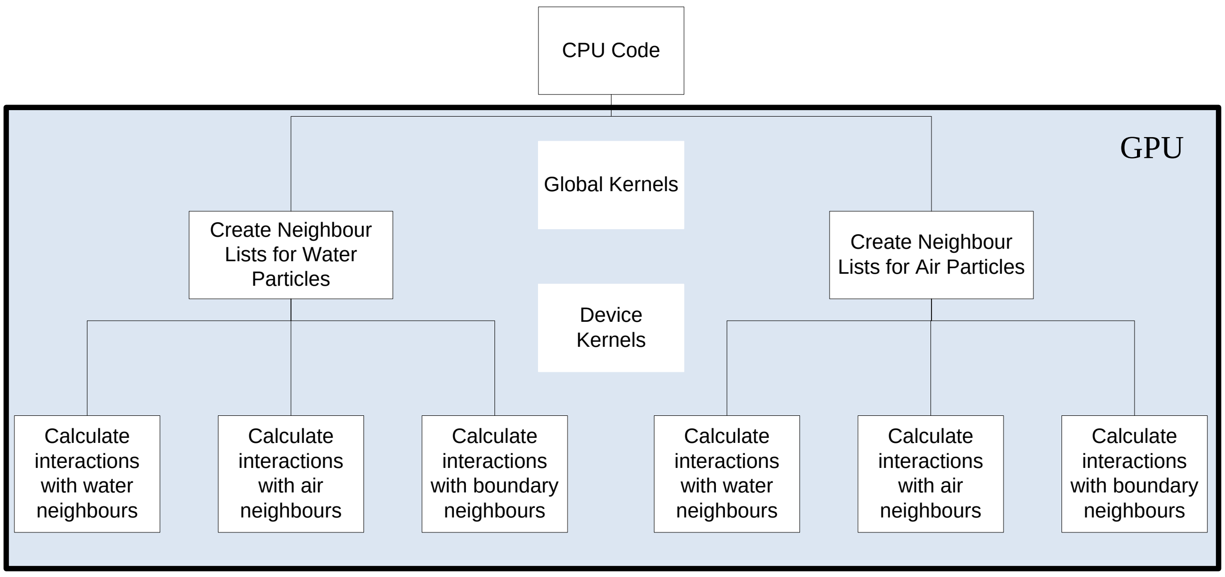

CPU Code
GPU
Global Kernels
Create Neighbour Lists for Water Particles
Create Neighbour Lists for Air Particles
Device Kernels
Calculate interactions with water neighbours
Calculate interactions with air neighbours
Calculate interactions with boundary neighbours
Calculate interactions with water neighbours
Calculate interactions with air neighbours
Calculate interactions with boundary neighbours

Figure 4.7: Global and Device Kernels for multi-phase DualSPHysics GPU code for Algorithm 3

This reduces the computational expense since the appropriate formulation is known *a priori* for each different phase-phase interaction, e.g. air-water and water-water, because the majority of conditional statements have now been removed. When the particles are interacting, we explicitly know the phase of each one due to the thread number of each particle and the different CUDA kernel used, removing the need for the particle ID filter of Algorithm 1 or the binary multiplier of Algorithm 2.

This approach is more complicated than Algorithms 1 and 2, with several new CUDA kernels and significant changes to the structure of the original DualSPHysics code. To reduce the computational runtime, the computational cost of launching new kernels should be less than the speed-up gained by eliminating conditional statements and reducing the complexity of the global kernels (e.g. for a two-phase flow one new global kernel and four new device kernels will be used).

Using multiple particle lists means that each particle list must be re-ordered every iteration. As mentioned previously, DualSPHysics uses the THRUST routine for particle list reordering to create cell and neighbour lists for efficient processing. In this new algorithm, the THRUST routine will now be called multiple times in each time step increasing the data handling but in theory leading to an improved performance. Furthermore, with the new global kernels called by the CPU host, there is more thread synchronisation that can potentially delay the execution.

A prerequisite for this approach is to ensure that the cell lists being created only contain particles of the same phase. Since the THRUST algorithm is still used for each phase, the access time for the particle data is also optimised for each phase, not for the whole computational domain. In fractured, violent domains this could be an issue near the interface with unequally formed cells that delay the construction of the neighbour list.

4.5.4. Algorithm 4: Intermediate CPU-GPU function

To avoid the time consuming CPU-GPU global kernel calls of Algorithm 3, a new function that connects the CPU and the global kernels is introduced. This new kernel minimises interaction between the CPU and GPU with only one global kernel called per iteration. The old global kernels are reconfigured as device kernels being called by the new, intermediate function solely on the GPU as shown in Figure 4.8.

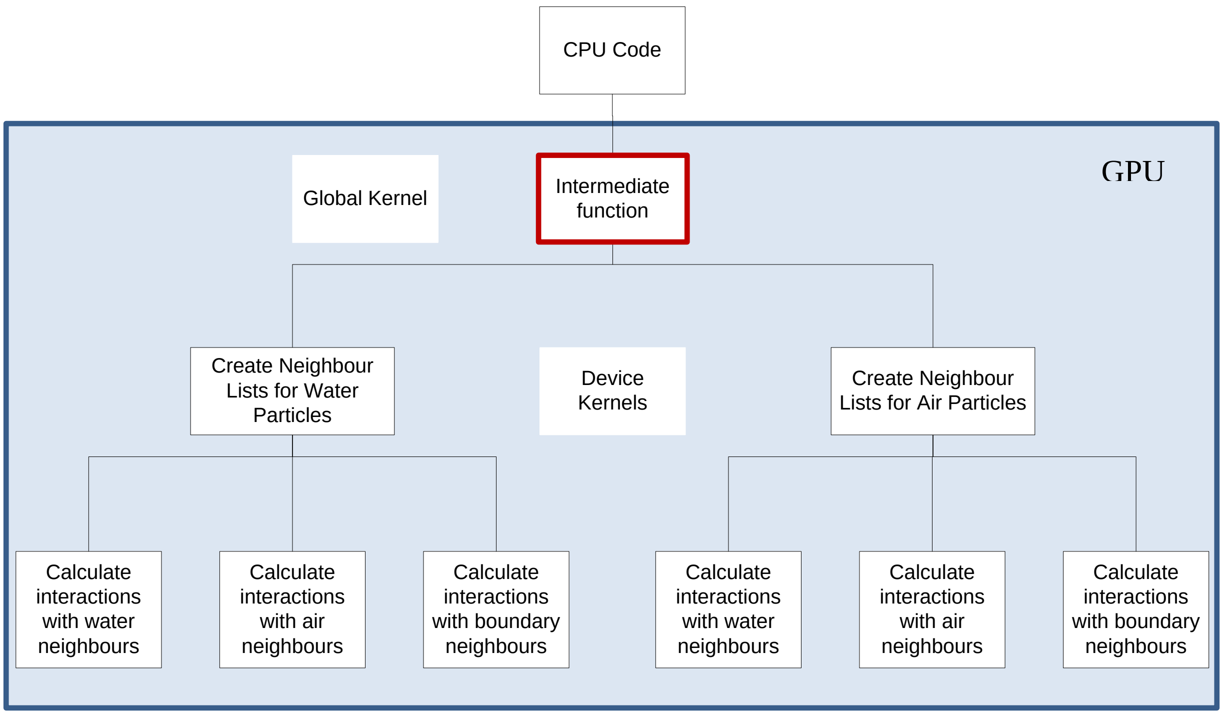

CPU Code
GPU
Global Kernel
Intermediate function
Create Neighbour Lists for Water Particles
Device Kernels
Create Neighbour Lists for Air Particles
Calculate interactions with water neighbours
Calculate interactions with air neighbours
Calculate interactions with boundary neighbours
Calculate interactions with water neighbours
Calculate interactions with air neighbours
Calculate interactions with boundary neighbours

Figure 4.8: Global and Device Kernels for multi-phase DualSPHysics GPU code for Algorithm 4

The major issue with this approach is the potential for increased code branching within the intermediate function. A new conditional statement is required in the intermediate function to separate the cell lists of the different phases, which were handled by different global kernels in Algorithm 3, as device kernels cannot assign separate computing resources for each phase. However, the internal structure of the old global kernels has not been modified from Algorithm 3, such that the particle lists for each phase with their advantages are retained. This leads to extra computational load for some threads and subsequent delays with synchronisation. This approach tests whether minimising CPU-GPU interaction is more important than minimising branching among the threads.

4.6. Hardware

Two different GPUs have been used to validate the GPU multi-phase code, an nVidia GeForce GTX570 and a Tesla S2050. Both are based on the Fermi architecture, but the Tesla S2050 is a dedicated scientific computation device, whereas the GeForce GPU is appropriate for computer graphics and gaming. The performance attributes of these particular cards are summarised in Table 4. The CPU used for the comparison is the Intel Xeon E5507, with 32GB of RAM and a clock speed of 2.27GHz.

	GeForce GTX 570	Tesla S2050
Compute Capability	2.0	2.0
Processor Cores	15(480 CUDA Cores)	14 (448 CUDA Cores)
Clock Rate (MHz)	1560	1150
Global Memory (MB)	1279	2687
Memory Clock (MHz)	1900	1550
Memory Bus Interface (bit)	320	384
Memory Bandwidth (GB/s)	152	148

Table 4: GPU Information and Statistics

The major difference between the GPU cards is the global memory available during computation as well as the size of the memory bus. With the Tesla card a simulation of 14 to 16 million particles can be performed depending on the algorithm used, while the particle limit of the GeForce card is 6.6 million particles.

4.7. Test Case 1: Still Water

4.7.1. Case Description

Here we use a simple test case of still water in a tank to investigate the speedup of the GPU multi-phase code while maintaining quiescent conditions. A volume of water has been placed inside a square vessel of dimension 4 m with a water depth of 2 m, filling exactly half of the domain. Above the water the tank is filled with air. The only external force applied to the tank is gravity. The domain used has the same dimensions as the one used for the next test case, the dam break by Koshizuka and Oka (1996). A sketch of the tank can be seen in Figure 4.9:

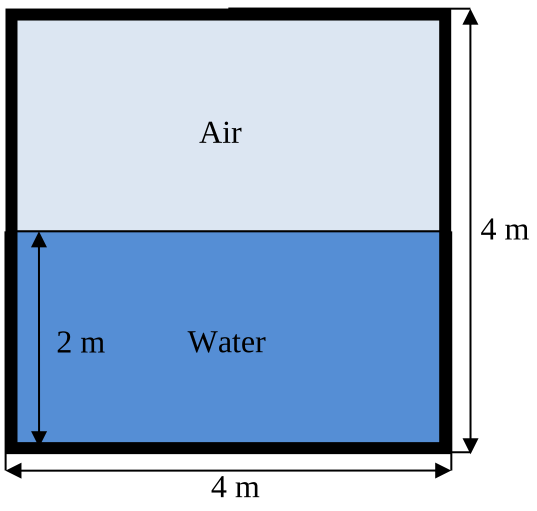


Figure 4.9: Definition Sketch for Still Water Multi-phase Case

This is a very simple case but an important one for Lagrangian methods regardless. Compared to the Eulerian formulation, mesh-free methods using a moving frame of reference struggle to maintain a zero-velocity domain. The still water case is then a suitable method for testing the algorithm to identify any problems or errors in the code. It is especially important in this case, due to the single precision used throughout the computation inherent to the GPUs. The case omits bottom friction in the tank, since this would damp any potential movement masking the errors present (Vacondio *et al.*, 2012).

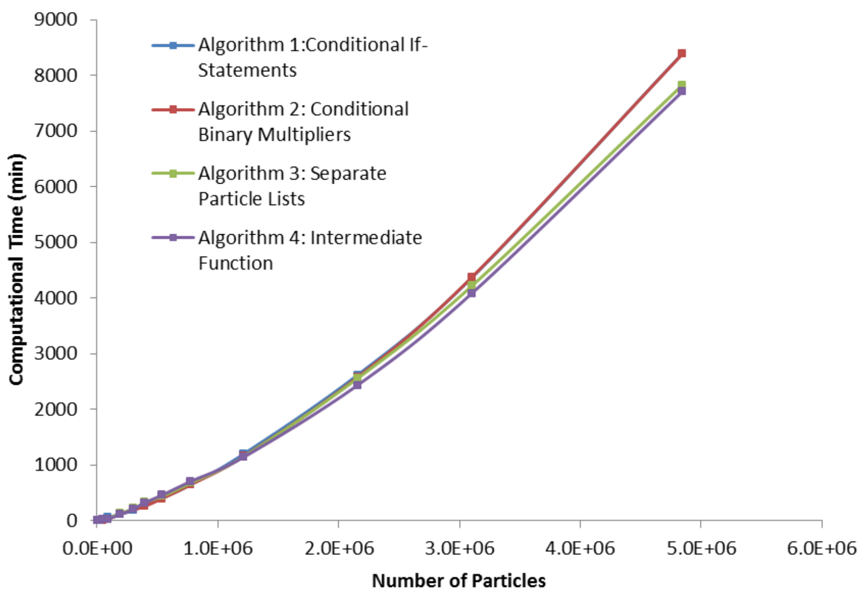
4.7.2. Comparison of different algorithms

This is a useful case for comparing the runtime of the different algorithms. The lack of fluid movement and the small forces present in the domain mean that the time step is almost exclusively directed by the relatively high speed of sound in the air phase and, as a result, remains nearly constant.

Figure 4.10 shows the computational time to run the simulation for 1 second of physical time where each algorithm was executed on both GPU cards for a range of particle numbers from 2000 to 12.5 million (note due to memory constraints, the GeForce card only simulates 4.5 million particles maximum). The results show that Algorithms 3 and 4, which use the separate particle lists, have the shortest computational runtimes and the difference among the algorithms is increasing when particle numbers are increased. When simulating 12.5 million particles, the greatest difference between the fastest and the slowest algorithms is approximately 8.5% of the simulation runtime.

Comparing Algorithm 1, which predominantly uses conditional if-statements, and Algorithm 2, which uses binary multipliers, Figure 4.10 shows only marginal differences in runtime. The faster simulation also depends on the number of particles, with Algorithm 4 being initially faster. The situation is reversed when further increasing particle numbers. Algorithm 3 has a more constant interaction between the CPU and GPU which will be independent of the number of particles. The intermediate function of Algorithm 4 uses limited code branching which may account for its greater runtime at 12.5 million particles.

(a) GeForce GTX570



(b) Tesla S2050

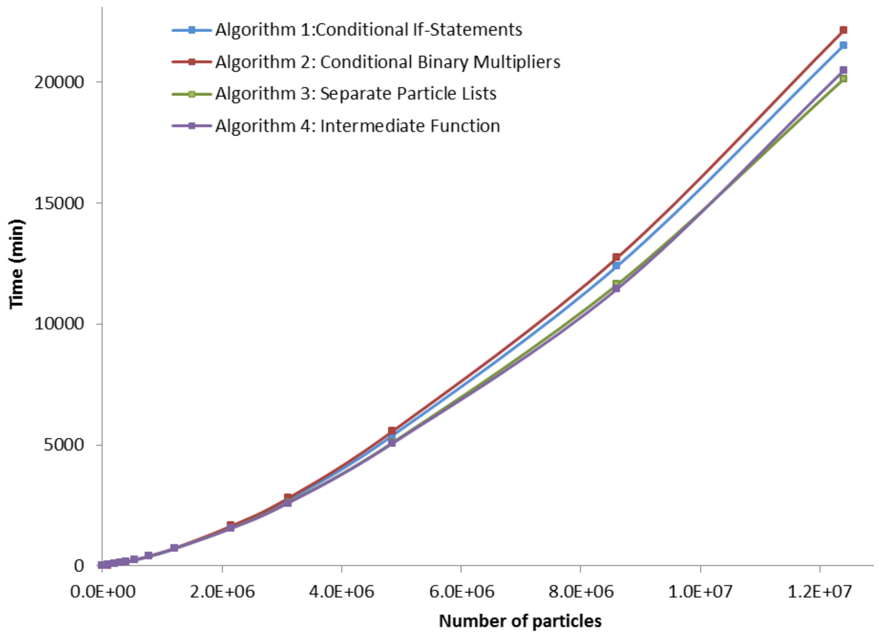


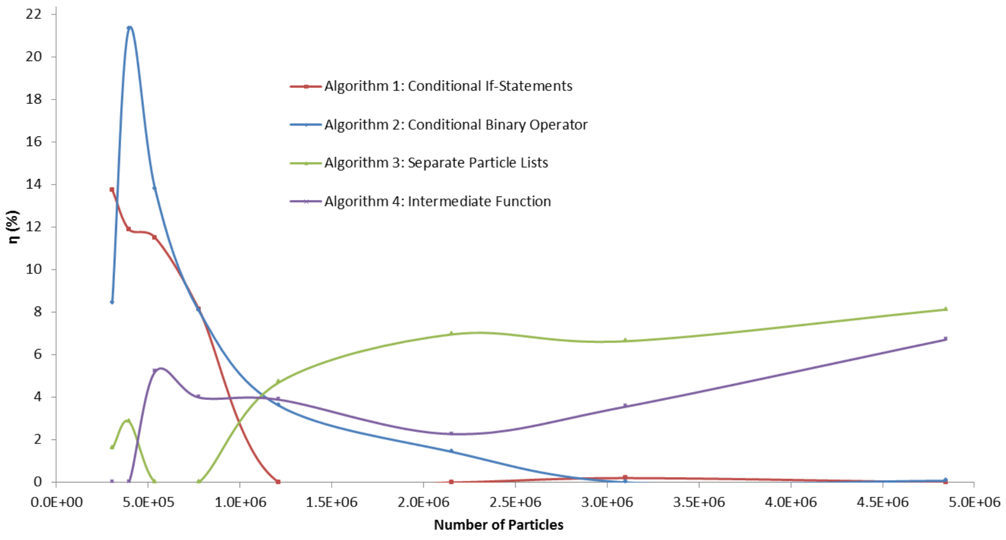
Figure 4.10:Runtime comparison of the different algorithms

The runtime difference for the algorithms has also been plotted in Figure 4.11. The runtimes are compared to the slowest algorithm for each simulation; Algorithm 4 for the low resolutions and Algorithm 2 for the higher. The results show that the time gained by using Algorithms 3 or 4 is on the order of 5-10% form Algorithm 2 and Algorithm 1 if more than 2 million particles are used. The runtime gain is computed using:

$$\eta = 100\frac{t_{slow} - t_{alg}}{t_{slow}}, \tag{4.1}$$

where t_{slow} is the runtime of the slowest algorithm, compared to the algorithm with runtime t_{alg}. The result is given as a percentage.

a) GeForce GTX570



b) Tesla S2050

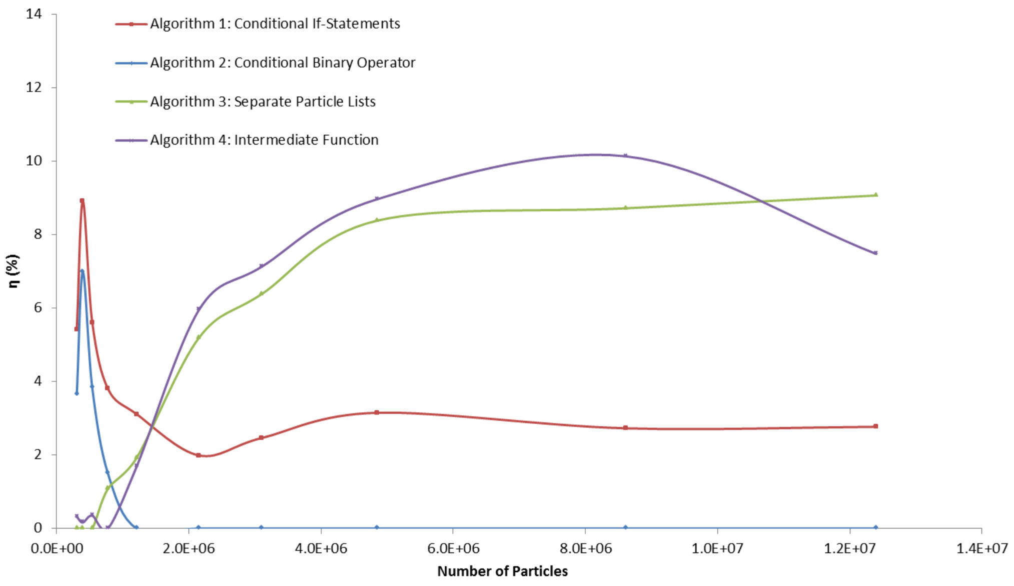


Figure 4.11:Runtime difference between the four algorithms

The results for the particle lists are confirmed when comparing the effect of the choice of GPU card as shown in Figure 4.12. Running the still water test case on the (slower) GeForce card gives a similar trend with Algorithm 3 being the fastest, even though GeForce simulations can only accommodate a fraction of the particles of the S2050. It is interesting to note that for lower numbers of particles, Algorithms 1 and 2 have a very significant advantage especially for the GeForce GTX570 card.

The percentage difference however does not mean the improvements gained by Algorithms 3 and 4 are insignificant. The runtime for the higher resolutions increases at an exponential rate for either card, as can be observed in Figure 4.12 resulting in the longer simulations taking days to finish. That increase means that any advantage in the higher resolutions potentially translates to several hours or even days needed for the computation.

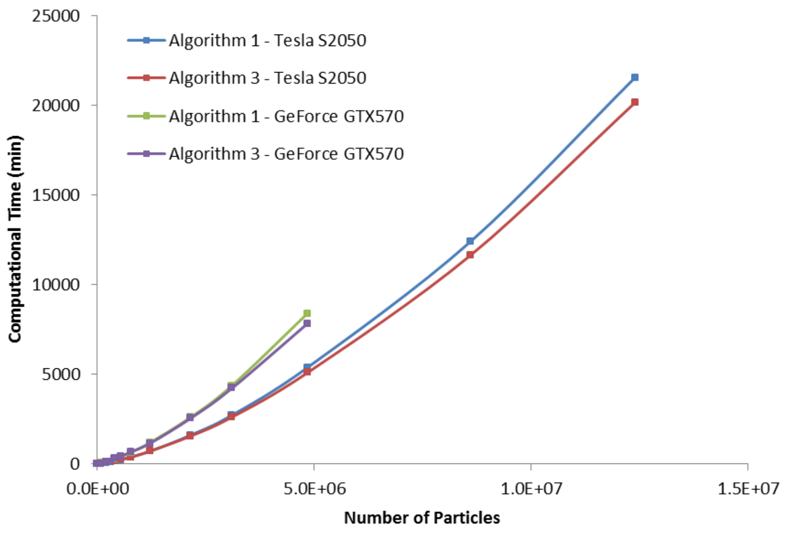


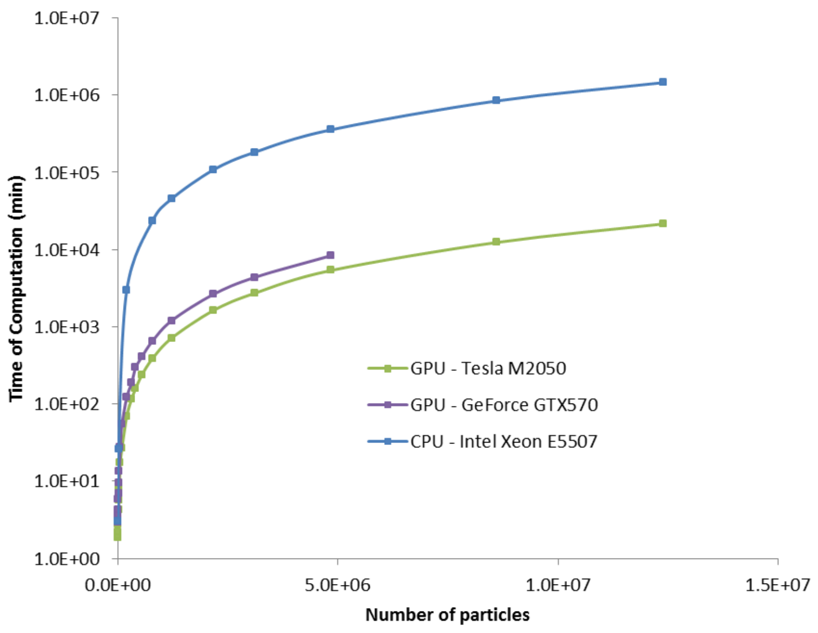
Figure 4.12:Runtime comparisons between the Tesla and GeForce graphics cards

Comparing the results of the GPU simulations to a CPU-only computation we find that there is a significant speed up gained as shown in Figure 4.13. The simulation was performed using different numbers of particles up to 12.5 million, which was the limit of the GPU memory. Figure 4.13(b) shows that the speed up for simulations for both GPU cards where the speed up is typically on the order of 40-80. The Tesla card has a speed up approximately twice that of the GeForce GPU which is not surprising given that the Tesla architecture is specifically designed for scientific computation.

As shown in Figure 4.13(b) the speed up provided by the GPU code for the fastest approach (Algorithm 3) is constantly increasing as the particle count is also increasing. There is a clear change in gradient in the speed up curves from 200,000 particles which is due to the increased time required for the particle interaction as shown in Figure 4.16. After 2 million particles the speed up remains constant, as the amount of data being processed negates the advantage gained from the GPU architecture.

This type of comparison is unfair for the CPU since it does not have the processing power of the GPU and because SPH is a method naturally oriented towards parallel computation. A more apt comparison would be to an MPI or an OpenMP code, which would also be parallel. However, this comparison is still useful as it shows the advantages of using a parallel approach as opposed to the conventional serial code.

(a) Computational time



(b) Speed up

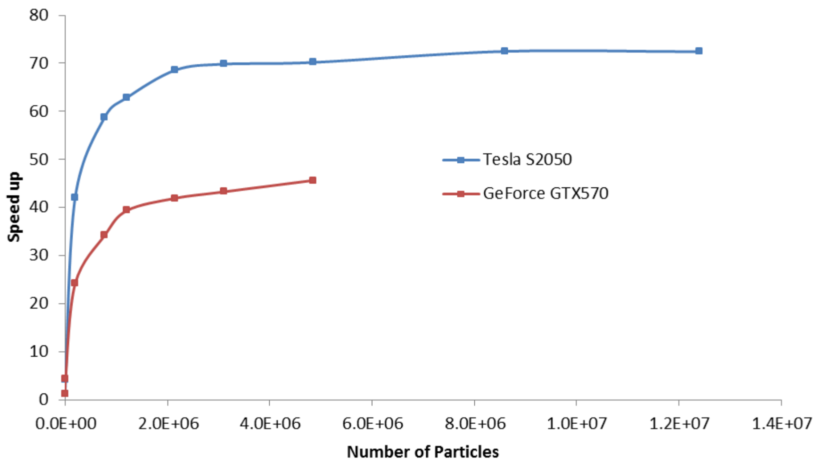
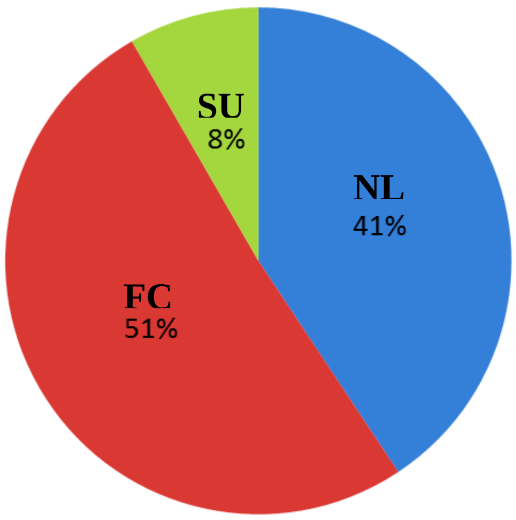
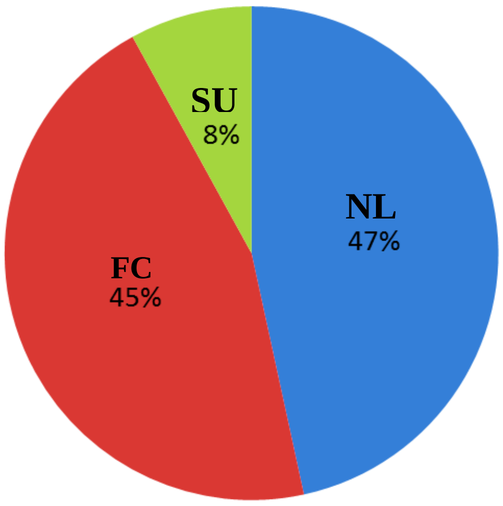


Figure 4.13: Runtime comparison between a CPU and two different GPUs

When considering the individual parts of the code, the creation of the neighbour list (NL), the force computation (FC) and the system update (SU), the difference between the computational times required to execute this task is better illustrated in Figure 4.14 and Figure 4.15 for 200,000 and 2,000,000 particles respectively. These figures show the percentage of the total computational time of the GPU that is devoted to each task.

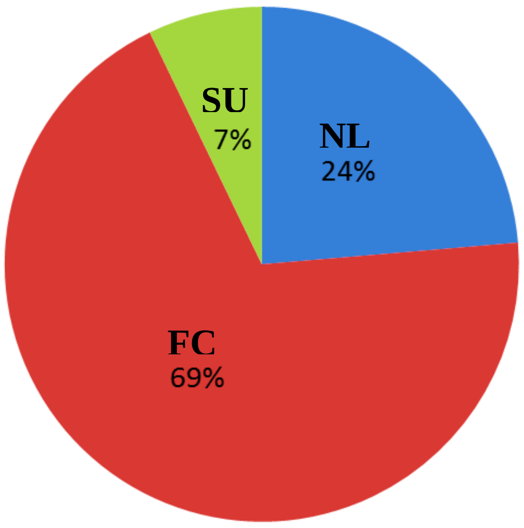


a) Algorithm 1

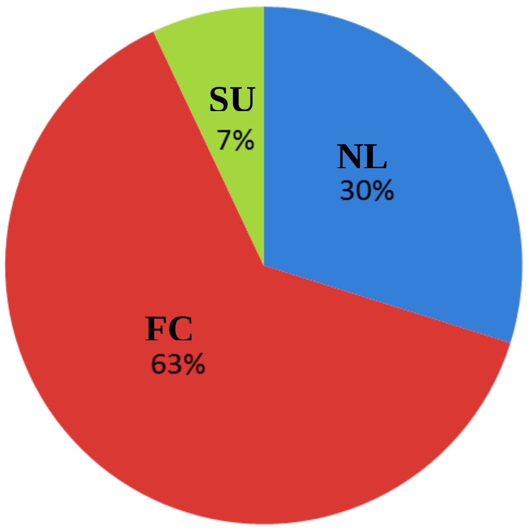


b) Algorithm 3

Figure 4.14: Percentage of the runtime taken by each part of the code for 200,000 particles (NL = Neighbour List creation, FC = Force Computation, SU = System Update)



a) Algorithm 1



b) Algorithm 3

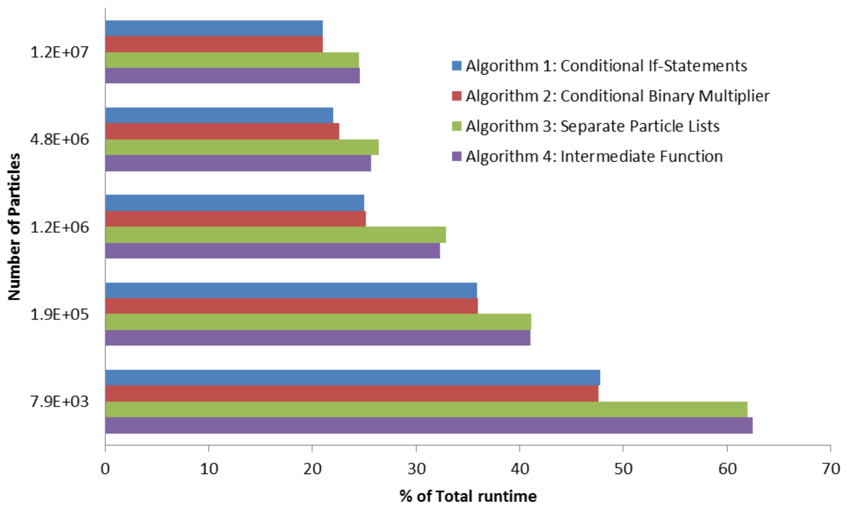
Figure 4.15: Percentage of the runtime taken by each part of the code for 2 million particles (NL = Neighbour List creation, FC = Force Computation, SU = System Update)

The particle interaction is indeed the most demanding aspect of the computation. Even at lower resolutions it requires close to 50% of the total simulation time as shown by Figure 4.14. The time allocated to force computation increases with the number of particles. This is not the case for the two other parts of the GPU computation. Updating the system variables at the end of each time step requires approximately the same computational effort regardless of the resolution or the algorithm, with minor differences at the order of 1%. On the contrary, the importance of the neighbour list function is diminishing, as shown in Figure 4.15 leading to an over 20% reduction on the time allocated.

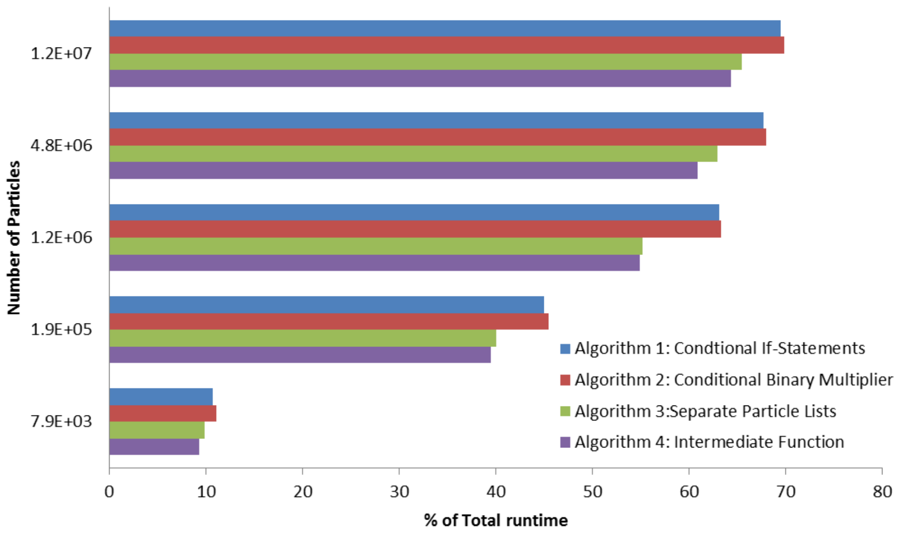
As shown in Figure 4.14 and Figure 4.15 between the two algorithms there is a difference in the allocated time for the neighbour list and force computation parts of the code. This difference is the same regardless of the resolution; the balance is shifted by 6% towards the creation and maintenance of the neighbour list for Algorithm 3. Figure 4.16 shows a more detailed description of the time devoted to each part as the particle resolution increases. It confirms the behaviour seen in Figure 4.14 and Figure 4.15: inter-particle force computation is the most demanding part of an SPH computation and the required computational resources are increasing with the number of particles.

The algorithms can be divided in two groups, depending on whether they use separate particle lists. The first group (Algorithms 1 and 2) which uses a single list devotes less time to the neighbour list and more time in the force computation, while the second group (Algorithms 3 and 4) has the exact opposite behaviour. The decrease of the time devoted to the neighbour list does not mean that the computational runtime for this part is smaller; as shown in Figure 4.17 both the parts of the simulation increase their runtime as the resolution increases.

(a) Neighbour List



(b) Force Computation



(c) System Update

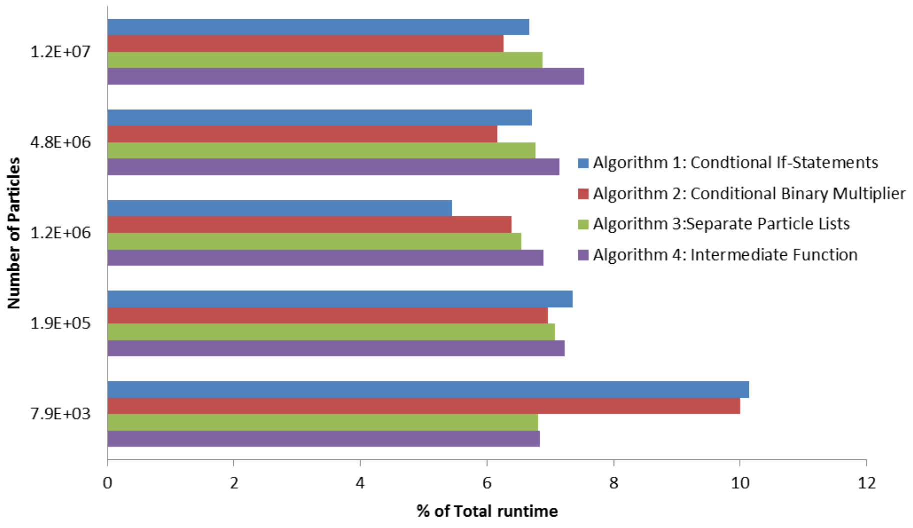
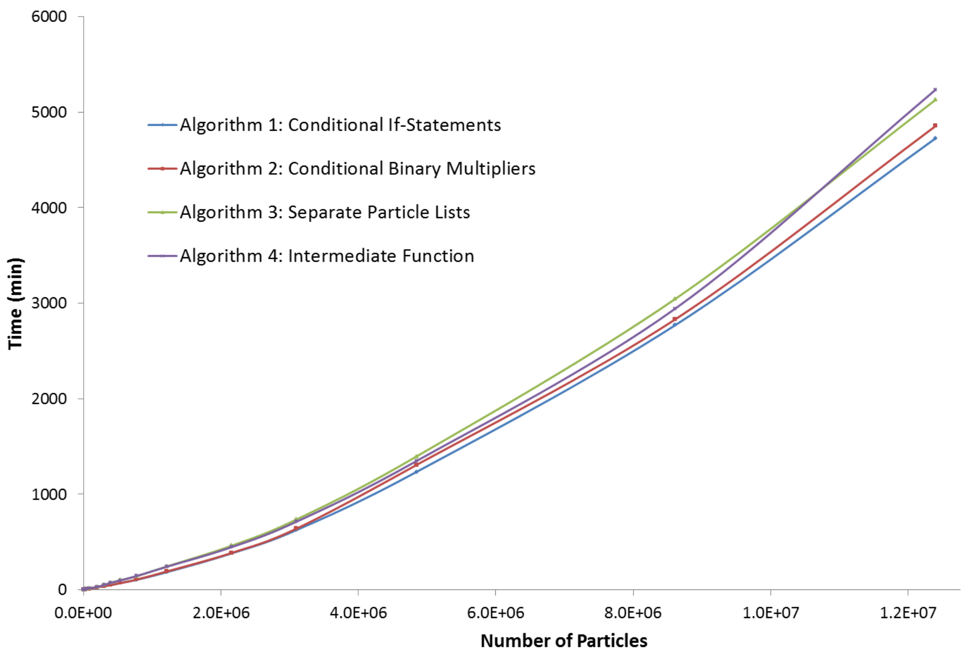


Figure 4.16:Comparison of neighbour list creation, particle interaction & system update for the different algorithms for simulations up to 12.5 million particles

a) Neighbour List



b) Force Computation

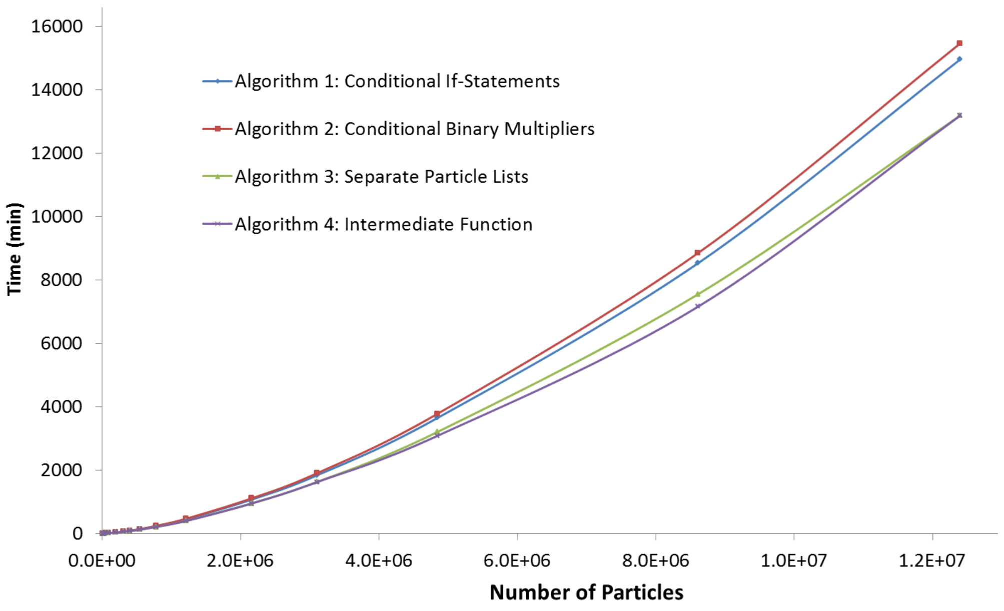


Figure 4.17: Runtime comparison of the computation parts for different algorithms

The difference in the required time for each part of the computation lies in their rate of change, with the runtime of the force computation increasing at a faster rate. Compared to the values of the neighbour list, this runtime is about 2-3 times larger. The difference in the percentage devoted to each part can be directly attributed to the runtime difference; Algorithms 1 and 2 require proportionately less time for the neighbour list and more time for the force computation. The first result is expected as they only require reordering one particle list in each time step. The second result occurs because of the elimination of conditional if-statements from the code. Launching separate kernels takes advantage of the inherent parallelism of the GPU, as opposed to separately checking the phase of each particle separately every time step.

This behaviour also offers an explanation for the difference between the runtimes of the various algorithms discussed herein. The creation of separate particle lists may increase the time required to create the neighbour list but the limiting factor of the computation lies in computing the particle interactions, so decreasing this runtime should be the main concern of any optimisation in the code.

The force computation limiting the speed of the simulation is further proven by looking at the profiling data for the GPU used in the simulation. These data were only available for the GeForce GTX 570 card. They show that the bottleneck for the system is the number of registers and blocks assigned to the *KerComputeFluid* kernel, which is the function that calculates particle interaction between fluid-fluid and fluid-boundary particles.

This kernel must be executed for every inter-particle interaction; for a simulation with 100,000 fluid particles, if a particle has for example 20 neighbours, this kernel needs to be executed 4 million times in each time step, since the predictor-corrector is a second order time integration model. Another limiting factor is the data accessed each time *KerComputeFluid* is executed. Limiting the number of particle interactions is impossible; however, having multiple particle lists allows limiting the amount of data being accessed in each execution cycle and a reduction of the commands being executed.

4.7.3. Comparison with the single-phase model

One of the greatest issues of applying a multi-phase model in SPH is the additional computational time required, due to the increased number of particles and the reduced time step. Depending on the model, additional equations and conditional statements for the treatment of each phase may be required and as shown in Section 4.5 creating an optimised algorithm is a difficult task.

As a result and as shown in Figure 4.18 the runtime of the multi-phase model is significantly larger than the single-phase. The comparison is between the original DualSPHysics code and the modified multi-phase version using Algorithm 3. Both cases are running an identical still water case, with the only difference being that the multi-phase code has to additionally simulate the air phase, so it has twice the number of particles. Both simulations have however, the same resolution and the same number of water particles.

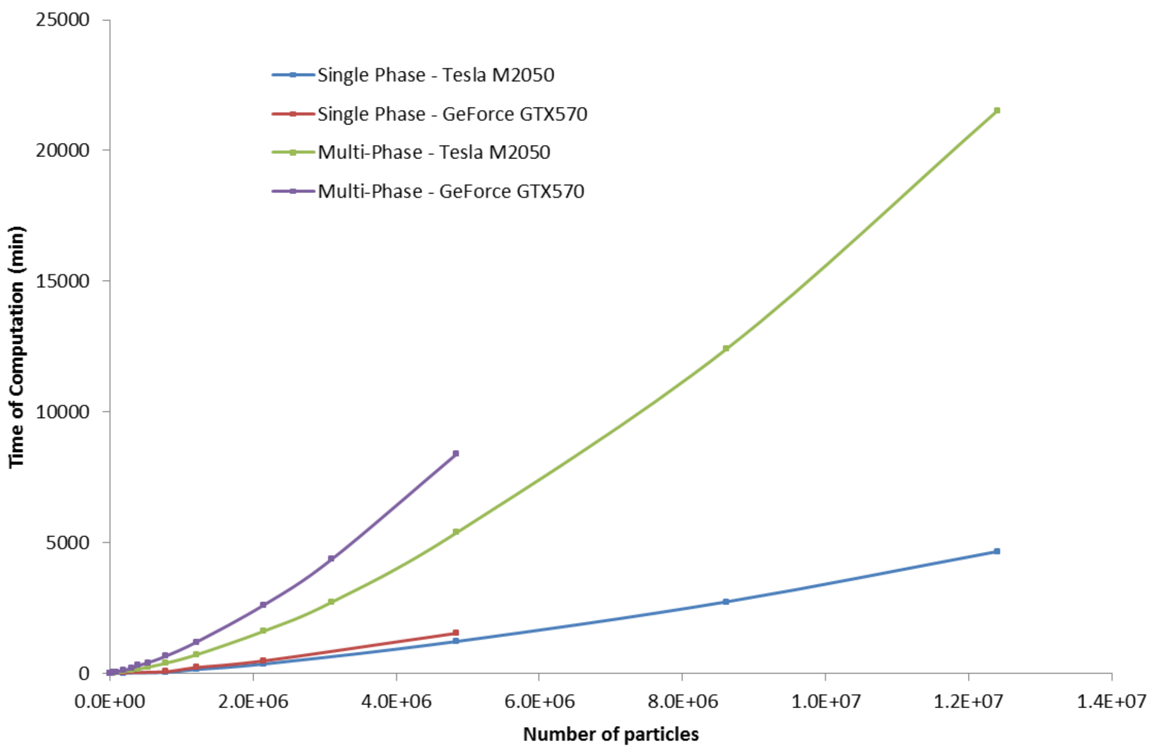


Figure 4.18: Runtime comparison of the single and the multi-phase computations for different GPUs

The difference is visible for every possible resolution with the single phase having a noticeable speedup which increases as the particle number is increasing. This holds true for both GPUs used in this study. A difference occurring is that the computational runtimes of the two GPUs are much closer for the single-phase code with the GeForce card being only slightly slower, while there is significant difference in the multi-phase simulation.

The previous observation is confirmed by the speedup, which is calculated to be around 4.2 for the Tesla card and 5.2 for the GeForce card. The difference can be attributed to the larger amount of registers for the Tesla card, which enable the bottleneck for the system, the *KerComputeFluid* kernel to be computed with a better efficiency.

With the previous comparison however, it is impossible to determine whether the difference is primarily occurring due to the additional equations for the air particles and the restrictions they impose on the time step or simply due to the increased number of particles. In order to determine the effect of these restrictions a simulation using a constant time step at 10^{-6}s has been executed. The time step was selected based on the restrictions imposed by the air phase so that both simulations will give correct and identical results. The runtimes for this simulation are presented in Figure 4.19 which compares Algorithms 1 and 3 with the single-phase simulation. The cases were executed using the GeForce 570 graphics card.

The simulation shown in Figure 4.19 uses the same number of particles for each simulation. The initial particle distribution is also the same for each simulation, meaning that any time difference occurs due to the additional treatment required for the air phase as well as the computational expense needed to separate the two phases.

The results show that the expense for treating the air particles is considerable; the single-phase simulation has a speed up around 2 compared to the two multi-phase simulations, as seen in Figure 4.20. Compared to the multi-phase simulation with the conditional *if*-statements (Algorithm 1) the speedup gained is constantly increasing although the rate of increase is slowing; it is possible that it will eventually reach a plateau, similar to Figure 4.13(b).

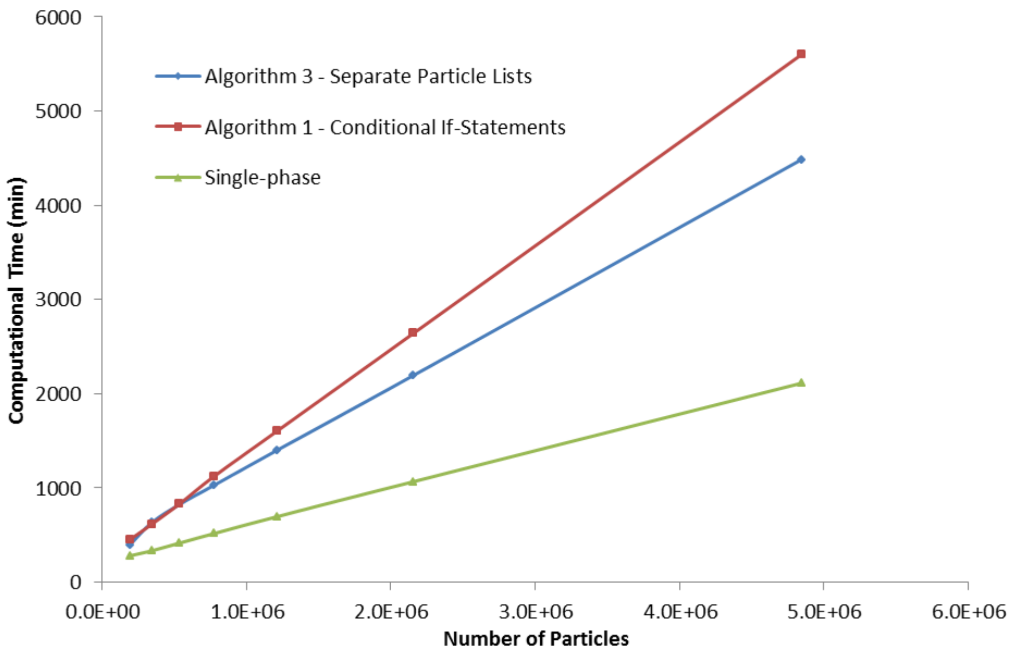


Figure 4.19: Runtime comparison of the results from Algorithms 1 and 3 with the single-phase simulation for a constant time step of 10^{-6}s an for the same number of particles

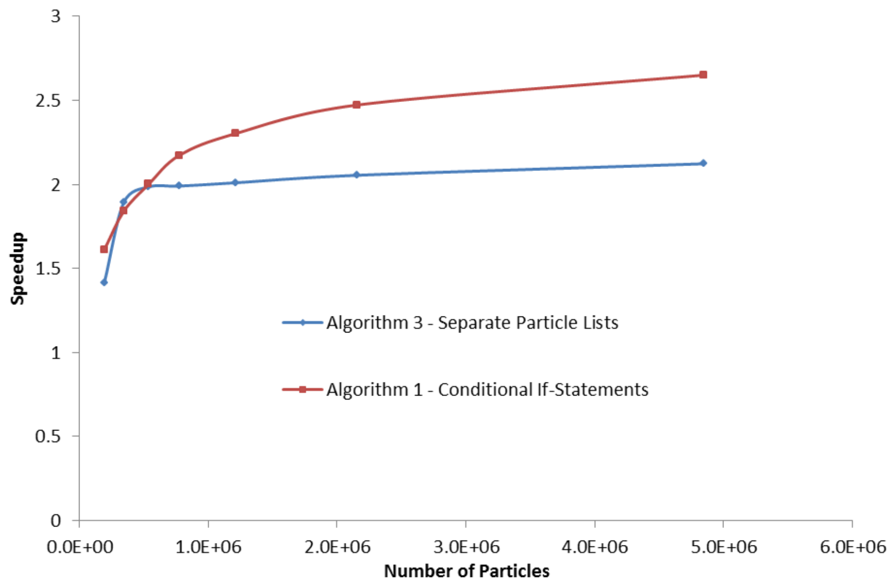


Figure 4.20: Speedup of the single-phase simulation compared with two multi-phase simulations for the same number of particles

The speed up gained by the single-phase simulation when compared to the simulation using separate particle lists (Algorithm 3) however, is much lower with the increase rate being effectively zero. The difference between the two algorithms illustrates that when using the same time step, applying separate particle lists for each phase leads to a lower computational runtime. This advantage is negated in a more complicated flow, where a variable time step is necessary to ensure the stability of the computation.

4.7.4. Extension to three dimensions

The multi-phase model used in this study has been introduced to DualSPHysics in a way that allows for an easy extension of the simulation to a three-dimensional space. This was also taken into account when designing the different algorithms. In that regard, it is essential to extend the still water simulation to the three-dimensional space. The still water simulation will expand in the third dimension as seen in Figure 4.21. The increase in the dimensions has been selected in order for the boundaries to not affect the majority of the fluid particles, while allowing for a high resolution to be used.

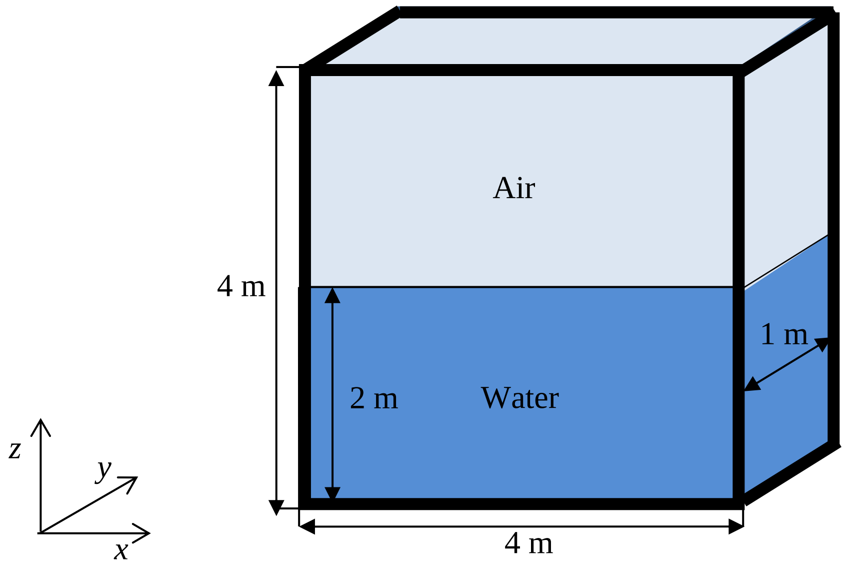
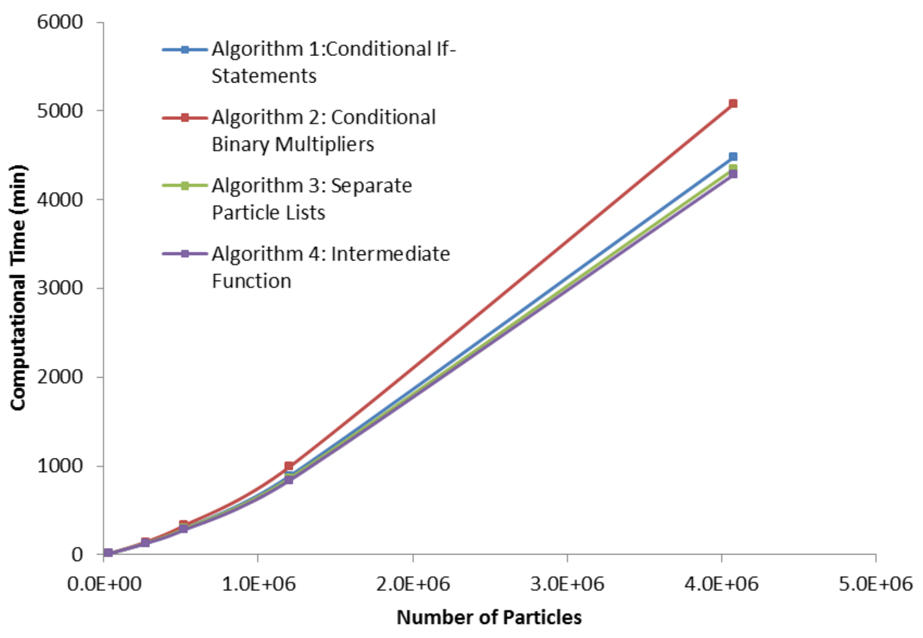


Figure 4.21: Speedup of the single-phase simulation compared with two multi-phase simulations for the same number of particles

Figure 4.22 shows the computational time to run the simulation for 1 second of physical time for this three-dimensional case. Each algorithm was executed on both GPU cards for a range of particle numbers from 30000 to 12 million in the Tesla card and due to memory constraints, from 30000 to 12 million particles in the GeForce card. The results are similar to the two-dimensional ones, presented in Figure 4.10 and show that Algorithms 3 and 4, require the least time to complete the simulation. The difference among the algorithms is increasing as the resolution is decreased.

The greatest differences appear when simulating 12 million particles. In that case the fastest algorithm (Algorithm 3) requires approximately 12% less time. Compared to the two dimensional case where the same difference was 8.5%, the benefit of separating the computation of the different phases is more apparent here. The difference is still not as noticeable for lower resolutions as can be seen in Figure 4.22(a)

(a) GeForce GTX570



(b) Tesla S2050

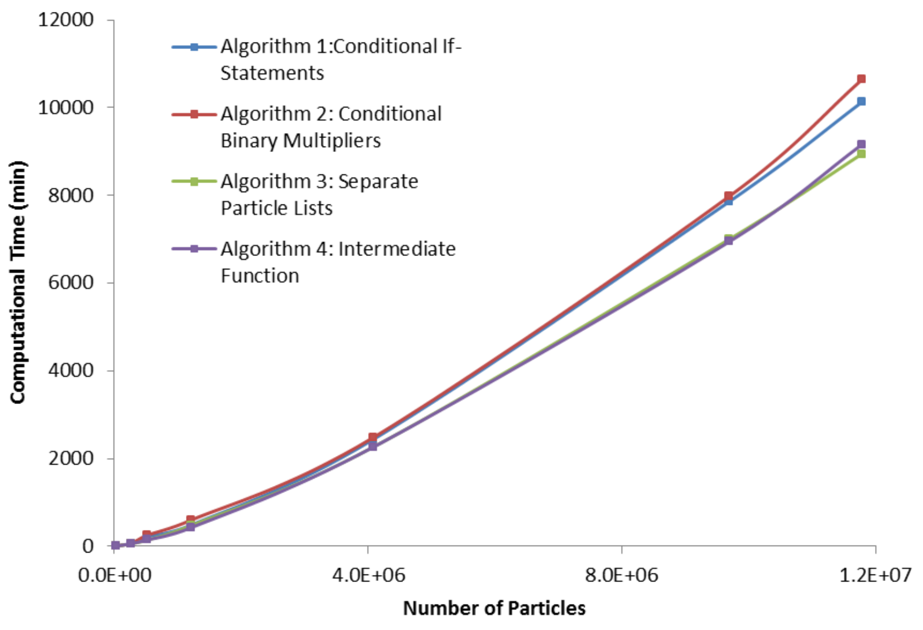
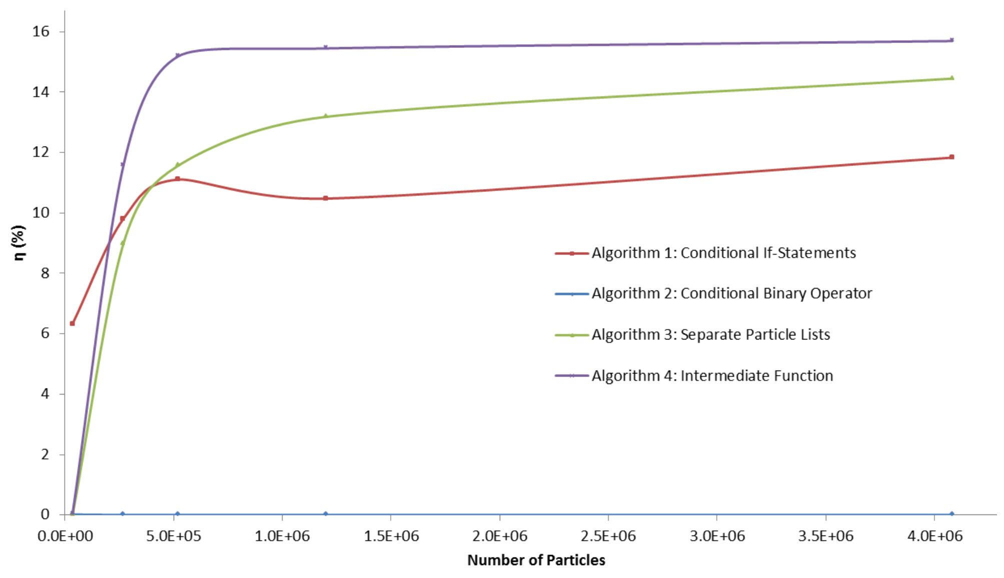


Figure 4.22:Runtime comparison of the different algorithms for a three-dimensional space

Regarding the computational runtime gained, Figure 4.23 shows that, unlike the results for the two-dimensional case, Algorithms 3 and 4 have lower runtimes regardless of the resolution used. Algorithm 2 is consistently slower, especially for the GeForce card, where Algorithm 4 is consistently the fastest. The speed gains are also larger compared to the 2-D case.

(a) GeForce GTX570



(b) Tesla S2050

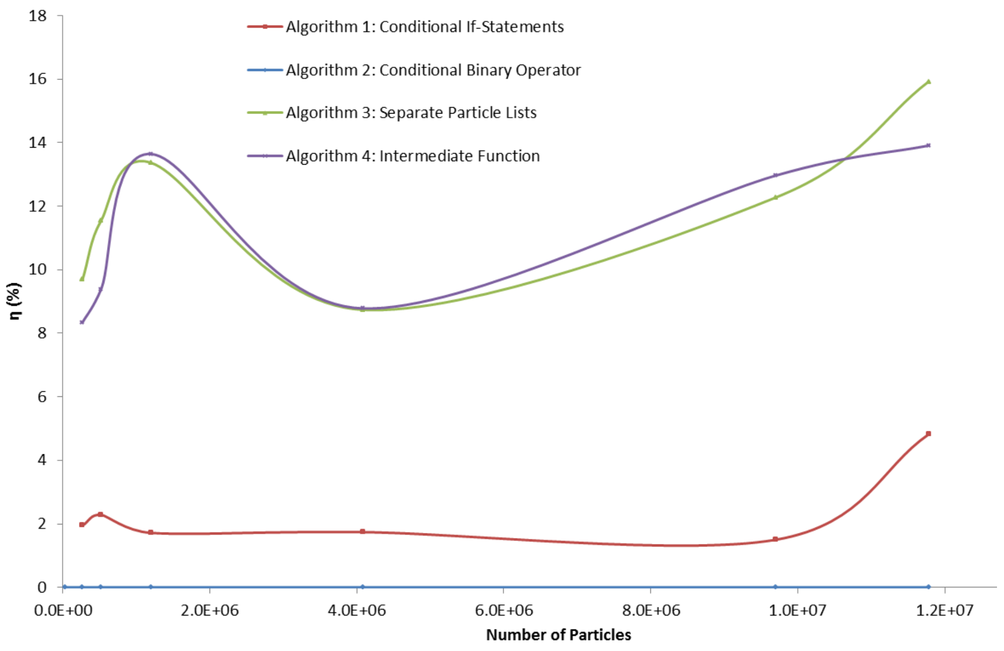
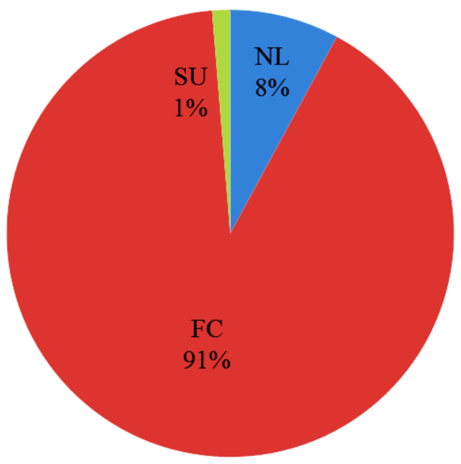
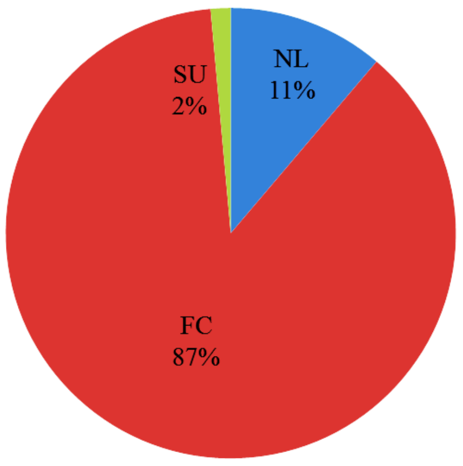


Figure 4.23:Runtime difference between the four algorithms for a three-dimensional case

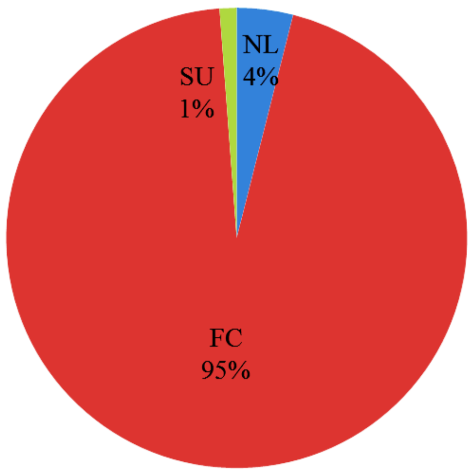
In order to investigate the increased difference between the two algorithms, the individual aspects of the algorithms will be investigated. Similar to the 2-D investigation we focus on the algorithms with the larger differences, 1 and 3 respectively. Figure 4.24 and Figure 4.25 show the percentage of time the GPU (in this case the Tesla S2050) devotes to completing each task. As with the two dimensional results presented in Figure 4.14 and Figure 4.15 the force computation is the most important part. It can also be observed that the creation of the neighbour list is a more important issue for lower resolutions and for Algorithm 3 which needs to maintain two lists.

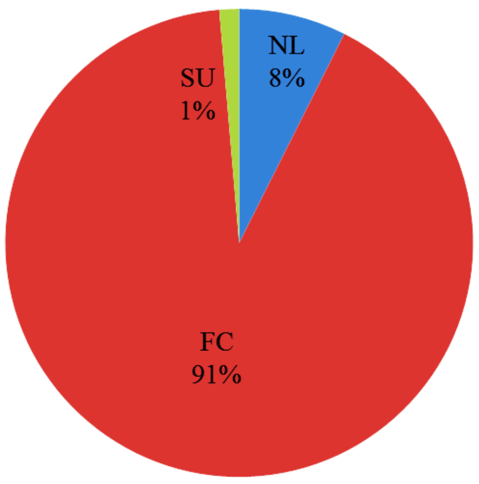




a) Algorithm 1 b) Algorithm 3

Figure 4.24: Percentage of the runtime of the three-dimensional still water case taken by each part of the code for 250,000 particles (NL = Neighbour List creation, FC = Force Computation, SU = System Update)





a) Algorithm 1 b) Algorithm 3

Figure 4.25: Percentage of the runtime of the three-dimensional still water case taken by each part of the code for 4 million particles (NL = Neighbour List creation, FC = Force Computation, SU = System Update)

Unlike the two-dimensional computation however, the force computation in this case now takes the vast majority of the time demanding over 90% of the total GPU computational time. This is a significant increase from the previous results, where the force computation required between 50 and 75% depending on the resolution. In contrast, the significance of the neighbour list and the system update is vastly reduced. For the two-dimensional case, creating the neighbour list requires a bigger investment of computational resources with the time allocated to this task increased by an order of magnitude (about 25-30% for a million particles depending on the algorithm).

Compared to the single-phase performance of DualSPHysics (Crespo *et al.*, 2011a), shown in Figure 4.26 the results are fairly similar with the time investment required for the neighbour list being increased. Consequently, the time devoted to the particle interaction is slightly diminished. The major difference between the two simulations is the increased number of cells being created as a result of the second phase which leads to an increased number of cell interactions. Note that the statistics shown in Figure 4.26 were calculated with a dam break simulation, where the particles exhibit significant movement and the contents of the cells are constantly changing.

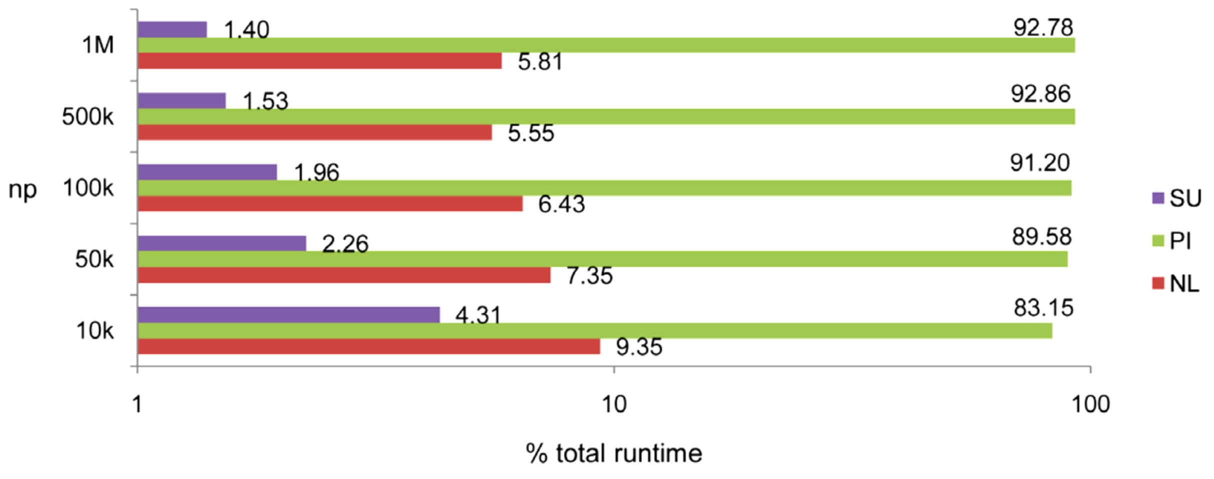


Figure 4.26: Computational Runtime Distribution in a Tesla M1060 (Crespo *et al.*, 2011a)

The reason for this change is the increased number of neighbouring particles, due to the fact that the smoothing length now affects particles in a sphere instead of a circle. This creates an increase in the number of particle interactions and the times the SPH algorithm is solved. The neighbour list and the system update algorithm on the other hand, only increase their computational runtime due to the number of particles without being greatly affected by the increase in dimensions.

The increased importance of the force computation part of the code also explains the increased difference between the two kinds of algorithms. As discussed in Section 4.7.2 and as shown in Figure 4.17, Algorithms 3 and 4 can calculate the inter-particle forces at a lower computational expense with dedicated particle lists for each phase. This is also confirmed by Figure 4.27 which shows the difference in computing the force computation subroutines. The difference observed mirrors the difference seen in Figure 4.22.

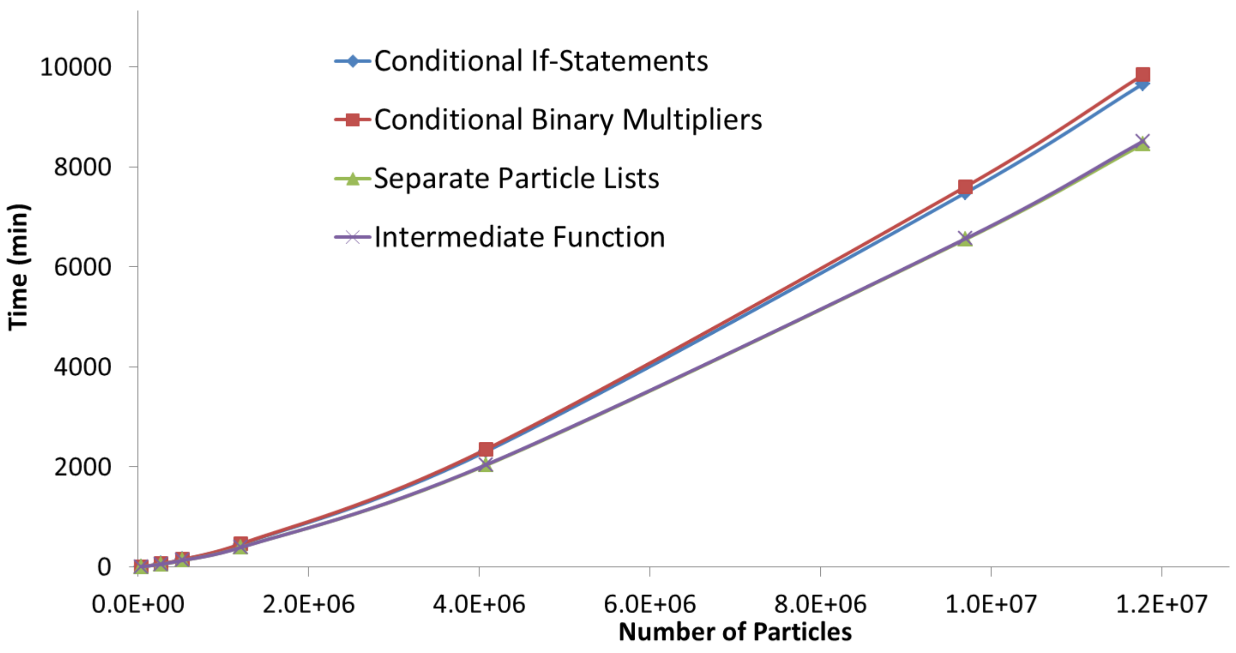
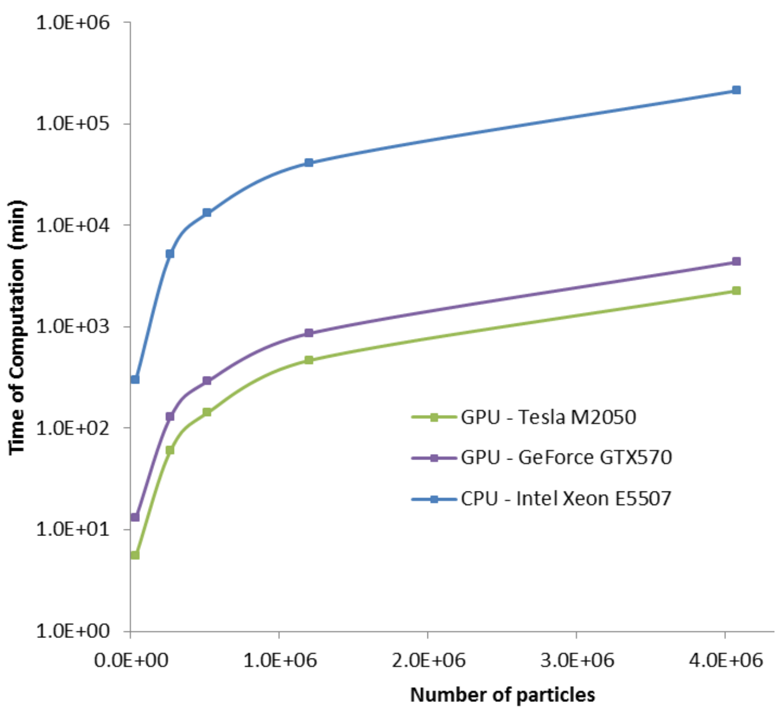


Figure 4.27: Runtime comparison of the force computation part of the code for different algorithms

When considering the comparison of the GPU to the CPU performance Figure 4.28(a) shows the difference in the computational runtimes, while Figure 4.28(b) shows that a significant speedup is obtained for both GPUs. Similar to the two-dimensional test case, the speedup gained is higher for the Tesla card and it reaches a plateau after about 1 million particles. Due to the increased importance of the force computation though, the runtime for the CPU simulation has increased disproportionately to the GPU one leading to a higher speedup that peaks at about 90.

(a) Computational time



(b) Speedup

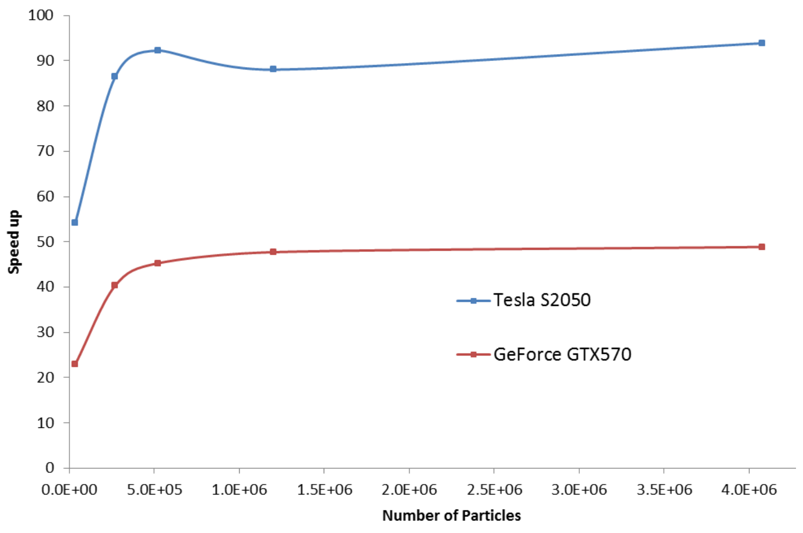


Figure 4.28: Runtime comparison between a CPU and two different GPUs for a three dimensional case

4.8. Test Case 2: Dry Dam Break

4.8.1. Case Description

The dam break case is a well-used benchmark for demonstrating the robustness of many SPH schemes and for testing its applications in impulsively-started, rapidly-evolving free-surface flows. A volume of water has been placed inside a vessel and flows only due to gravity, with no other external forces applied. The water is interrupted by the vertical wall located downstream with the resulting impact forcing the water to flow on the opposite direction. The reflected wave can, depending on the velocity of the flow, create air pockets within the water flow which significantly affect the resulting impact force and behaviour of the wave (Peregrine and Thais, 1996).

This is a simple case, but several hydrodynamic problems are directly related to its principles, especially wave impact problems. The latter include a diverse array of circumstances, from slamming loads on ship hulls impacting on the water surface (Faltinsen *et al.*, 2004), to water-wave impacts on walls and coastal structures (Peregrine, 2003) to sloshing problems in partly-filled tanks (Faltinsen *et al.*, 2000). A case similar to the latter will be investigated in Section 5.6. Of particular interest in these problems is the impact force of the waves on the structure, as well as the water pressure and the effect of the entrained air.

Figure 4.29 shows a definition sketch where the dimensions use the same length scale as the experiments performed by Koshizuka and Oka (1996). The actual dimensions are in line with the computations performed by Rogers *et al.* (2009). The water is placed inside a completely closed square vessel with 4m walls. There is no void inside the vessel with the remaining space occupied by air particles. This is a dry dam break case without any water at the bottom of the domain and the movement of the gate does not affect the water flow. As a result the gate is considered to be removed immediately at t=0 without affecting the flow.

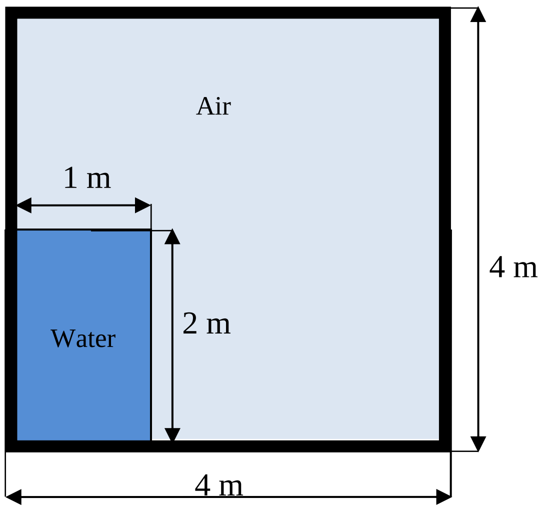


Figure 4.29: Definition Sketch for Dam Break Multi-phase Case

4.8.2. Runtime results

The dam-break test-case has been simulated with different particle resolutions. This case displays significant water movement and has, as a result, a variable time step. Figure 4.30 shows the computational time to run the simulation for 1 second of physical time for the different algorithms. The simulation was performed in the Tesla M2050 card. Algorithms 3 and 4 have a slightly lower computational time, confirming the results from the still water test case. The differences among the different algorithms however, are relatively small although not insubstantial, especially for the higher resolution.

The results from the still water case are also verified when comparing the results on different hardware. Figure 4.31 shows the difference in runtimes between the Tesla and GeForce GPU cards. The dedicated architecture of the Tesla card offers a significant decrease in computational runtime regardless of the case, while the increased amount of memory available also allows for the simulation of a larger number of particles.

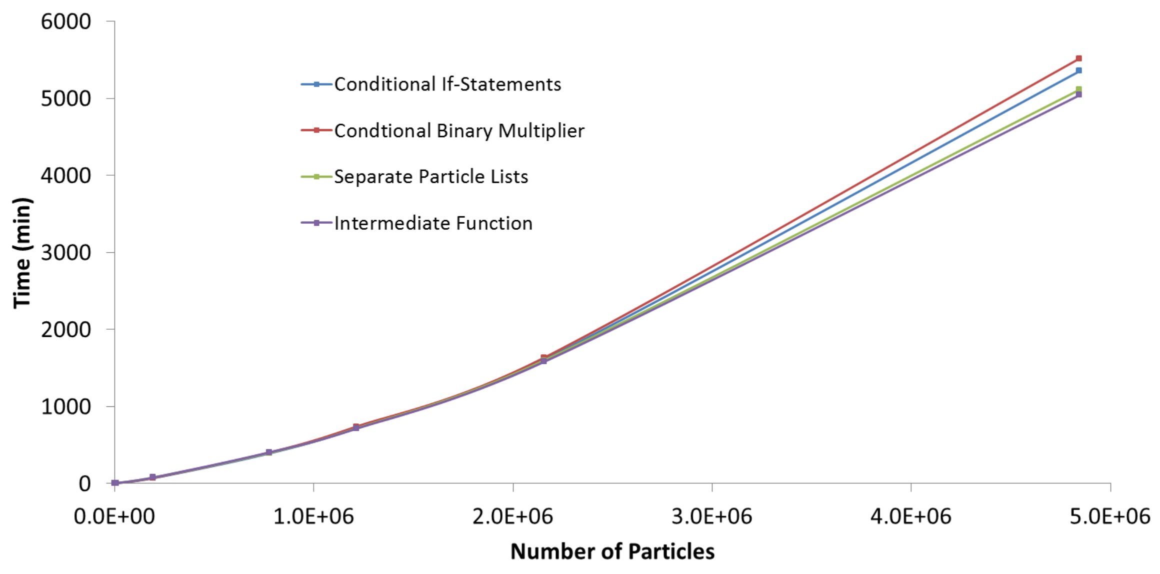


Figure 4.30: Runtime comparison of the different algorithms for the dam break test case

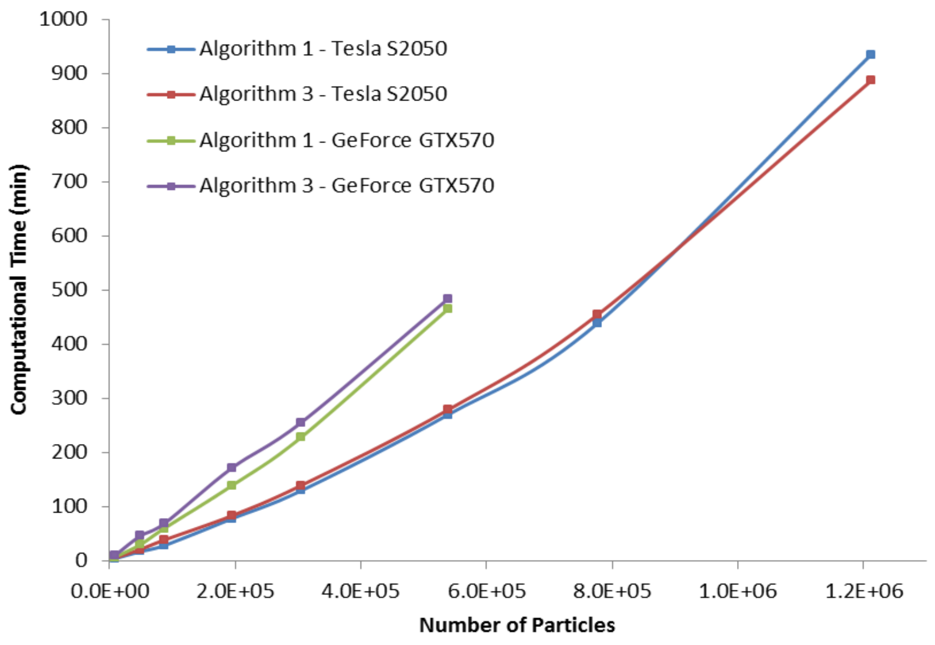


Figure 4.31: Runtime comparison of Algorithms 1 and 3 for different hardware for the dam break test case

In comparison to the still water case shown in Section 4.7, the computational speed is slower. The decreased time step is occurring as a result of the increase of the water flow velocity and as a result of its interaction with the opposite wall, when the forces and pressure are rapidly changing. The increase in inter-particle forces means that the time step is limited by Equation (3.54) instead of Equation (3.55), which is mostly affected by the speed of sound in the air phase and is the dominant restriction for the still water test case. Figure 4.32 shows the difference between the two cases for 1 s. of physical time.

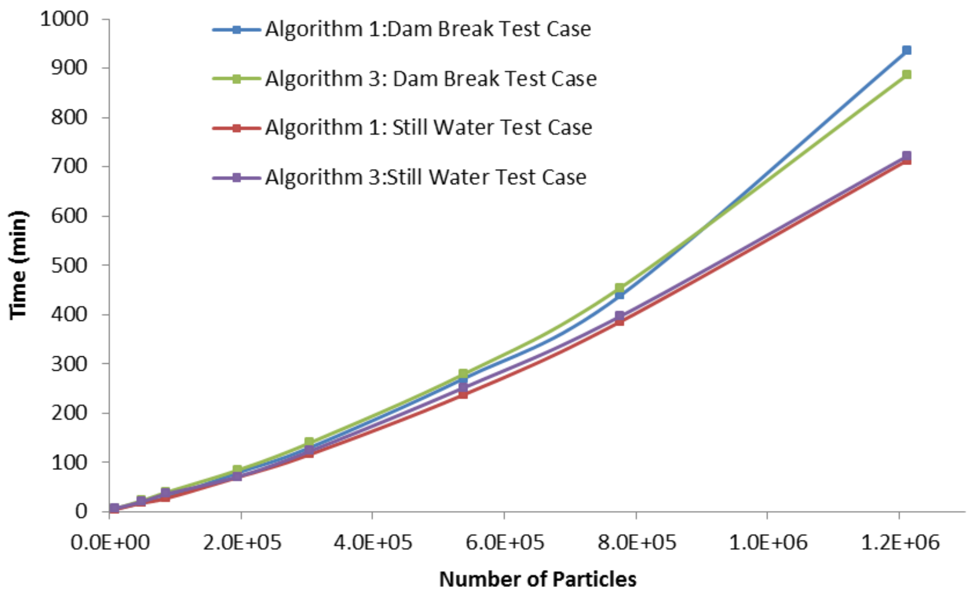
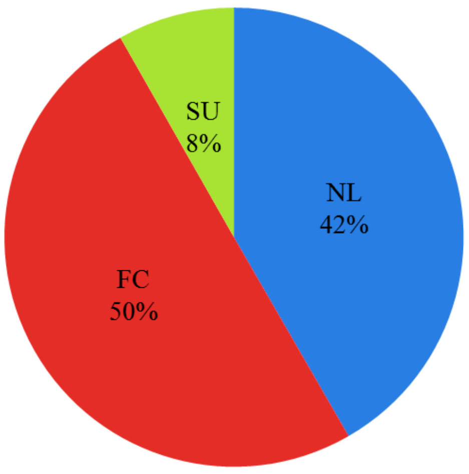


Figure 4.32: Runtime comparison of the dry dam break and the still water test cases for Algorithms 1 and 3

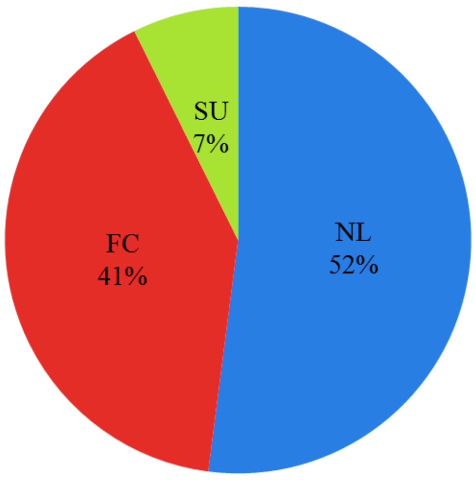
The difference in runtime is becoming significant after simulating 800,000 particles and the effect is increased when moving to higher resolutions. The difference between the algorithms is more pronounced algorithms in the dry dam break while in the still water case the algorithms have approximately the same runtime. Using Algorithm 1 is slightly faster, but the situation is reversed for the dry dam break, where Algorithm 3 has a lower runtime and the difference is more noticeable.

The difference in the size of the time step depending on the point of the computation means that identifying the average time step is a more difficult process and it will be different depending on the length of the simulation. Indeed, simulating the overturning wave appearing at the later stages of the dry dam break case requires a lower time step due to the complexity of the flow.

When considering the individual parts of the code, the percentage of the time taken is given by Figure 4.33 and Figure 4.34 for 2×10^5 and 2×10^6 particles, respectively. The trend follows the one presented in Figure 4.14 and Figure 4.15 for the still water case: the force computation is the most time-consuming part of the simulation for higher resolutions and the neighbour list. The differences between the two cases are very small, in the range of 1%, so the conclusions for the still water case can also be applied to the dam break case.

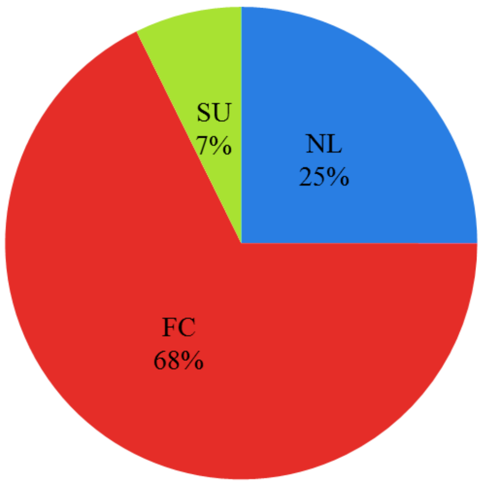


a) Algorithm 1

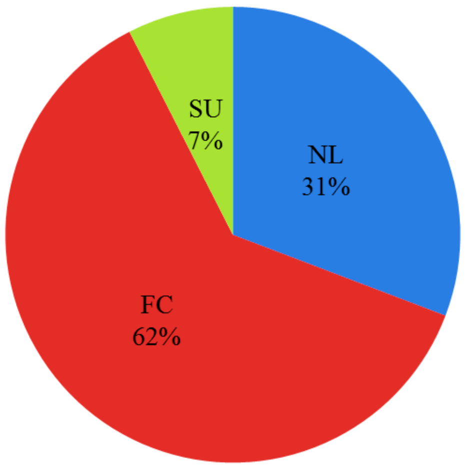


b) Algorithm 3

Figure 4.33: Percentage of the runtime of the dam break case taken by each part of the code for 200,000 particles



a) Algorithm 1



b) Algorithm 3

Figure 4.34: Percentage of the runtime taken by each part of the code for 2 million particles

4.9. Concluding Remarks

In this chapter, a description of the code being used in this study was given along with the improvements and modifications executed in order to incorporate the multi-phase model. Several algorithms were proposed for the optimal interaction between the phases, each with a different focus. Testing these algorithms and their effects on the computational runtime of the code shows an advantage if the CUDA kernels computing the particle interaction are separated from each phase. The advantage is greater as the number of particles is increasing. The next chapter will investigate the application of the shifting algorithm for the multi-phase model as well as using the code for simulating some 2-D test cases such as a dam break (wet or dry) or a rolling tank.

5. Validation: Two Dimensional Cases

5.1. Introduction

The next two chapters will present several cases used for validation of the multi-phase GPU code. Test cases that would be very time consuming without a GPU code are investigated. This chapter will cover the two-dimensional cases, while the next chapter will present a three-dimensional case. At high particle resolutions, voids appear in the air phase. A shifting algorithm is introduced to counter these issues. The code is validated using both wet and dry bed dam break cases comparing them to experimental data. The evolution of a rolling tank, partly filled with water will also be simulated. Due to the presence of air, and to reduce the computational demands, the cases presented here are completely enclosed by boundary particles.

5.2. Dry Dam Break

5.2.1. Case Evolution

The dam break case evaluated here has already been presented in Section 4.8 using data from the experiment of Koshizuka and Oka (1996) and concerns the evolution of the collapse of a water column over a dry bed. This case was selected because of its simple geometry and flow mechanics; as a gravity-driven flow little difference is expected when changing parameters. It will allow however, determining whether the water movement is simulated correctly, or whether it is significantly affected by the presence of the air phase.

After the start of the simulation the water column collapses under the effect of the gravity and constantly moves towards the right side of the domain, where, about 0.7s after the beginning of the simulation, the flow collides with the opposite wall reducing the velocity of the toe particles to 0. Figure 5.1a and Figure 5.1b show two instances of the simulation at 0.4s and 0.7s respectively. Following the wall interaction and as a result of the forces exerted by the moving part of the flow, the toe position moves upwards as seen in Figure 5.1c.

a) The air phase only plays a minor role during this part of the simulation. The gravity flow created exhibits low velocities before reaching the opposite wall leading to minimal mixing between the air and the water phase. The interface is constantly changing but this is not a violent flow; the free surface remains intact. Only when the water toe reaches the right edge of the domain are some air particles separated from the main air flow, as seen in Figure 5.1c in the bottom right corner.

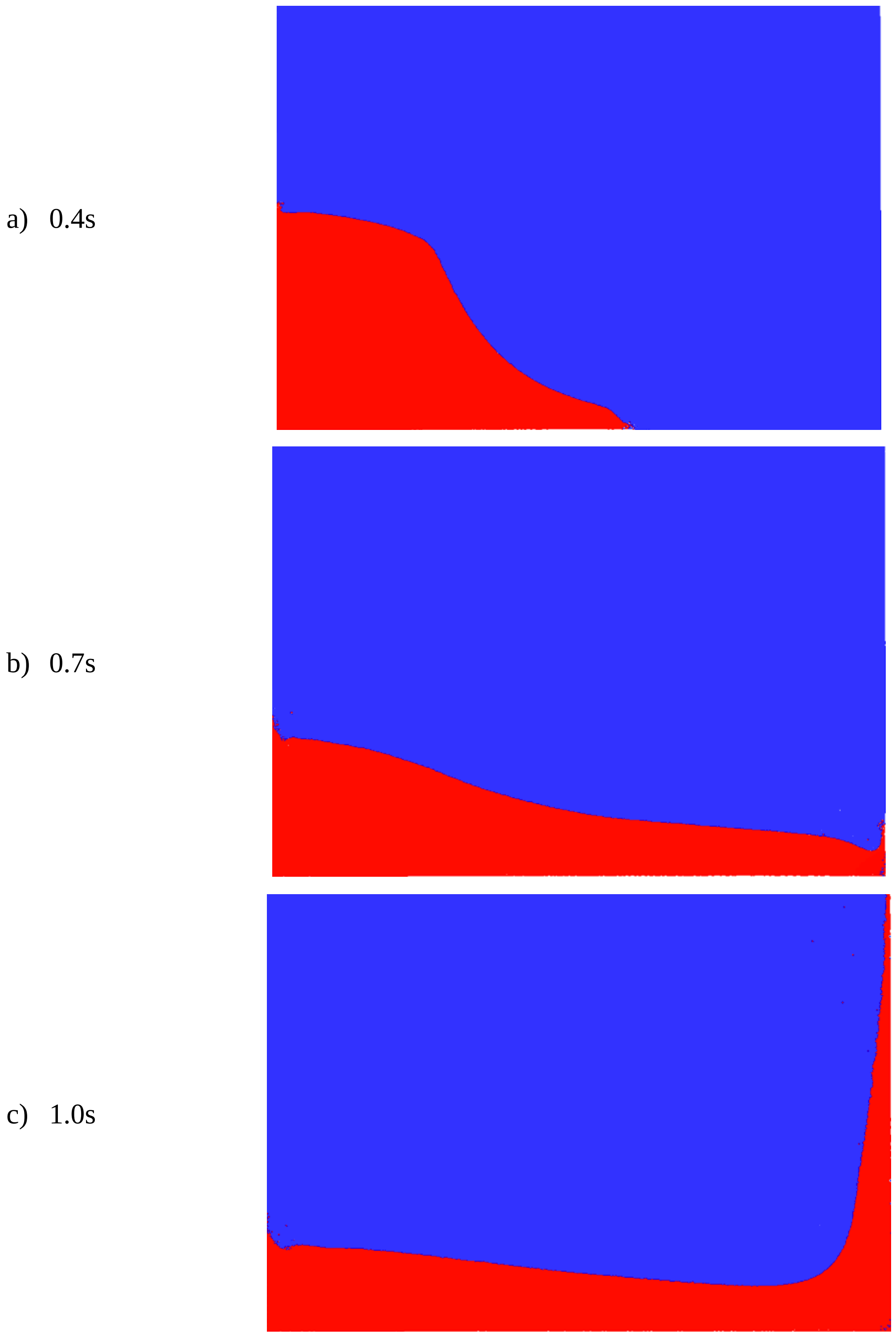


Figure 5.1: Instances of the dry dam break case taken from a simulation with 250,000 particles

The dam break case also demonstrates some issues occurring from the use of the dynamic boundaries (Crespo *et al.*, 2007):

a) Some water particles close to the left wall have a delayed movement, separating from the main flow. The effect is noticeable in the instances of Figure 5.1, especially in Figure 5.1(b) and Figure 5.1(c) in the bottom right corner. The forces exerted by the boundary particles prevent their downward movement due to the way they are positioned. This force is present in the other fluid particles near the interface, however, in this case, the incomplete kernel and the very low velocity of these fluid particles amplifies the error. Repulsive forces exerted by the air particles are also significantly lower than particles with a high density, so the force equilibrium does not make the water particles follow the main flow.

b) After the water column begins collapsing the distance between the boundaries and the water particles increases to 1-2 times the smoothing kernel length *h*. It is more noticeable in Figure 5.1(a) at the bottom near the toe of the water flow. The jump occurs due to the small number of water particles in the flow toe. The repulsive forces from the boundary particles overcome the repulsion from the air phase and the water flow moves slightly upwards till equilibrium is maintained.

5.2.2. Validation

When executing the experiment (Koshizuka and Oka, 1996), the authors were interested in measuring the toe position of the wave and the height of the flow. The position of the toe was measured until the water reached the opposite wall, while the height is the depth of water at the left wall, where the water flow was located initially at t=0 and it is measured for the first 1.2s of the simulation. The reason for measuring these two quantities is to assess the velocity with which the water spreads and the rate of fall for the water column (Martin and Moyce, 1952).

To compare the results to the Koshizuka and Oka (1996) experiment, the length and height of the water flow as well as the computational time have been non-dimensionalised according to the following equations (Rogers *et al.*, 2009):

$$\text{Toe Position}\quad X^* = x / h_0, \tag{5.1}$$

$$\text{Water Height}\quad H^* = h / 2h_0, \tag{5.2}$$

$$\text{Computational Time}\quad t^* = t\sqrt{2g / h_0}\ , \tag{5.3}$$

where h_0 is a characteristic length of the case, selected here to be equal to the initial height of the water column (Koshizuka and Oka, 1996). The particle spacing dx used in this case can be non-dimensionalised with the initial height of the water column h_0. For the simulation presented, the value used is dx/h_0=0.008 while the speed of sound for the water has been selected in order to ensure that the Mach number is less than 0.1 (Monaghan, 1994). For this case it can be found using the Equation:

$$c_{sw} = 10\sqrt{gh_0} \,, \tag{5.4}$$

while the speed of the sound for the air phase can be found by the Equation (3.61). The Reynolds number can be calculated using the initial height of the water column as characteristic length, while the maximum propagation speed can be found by $\sqrt{gh_0}$ so the value of the Reynolds number is ~3×10^6.

Figure 5.2 shows the results for the non-dimensionalised toe position while Figure 5.3 shows the comparison for the height of the water column. The results are compared to a single-phase simulation using the DualSPHysics code with the same particle size.

The toe position results are also compared to the analytical solution by Ritter (1892). The solution considered was a horizontal channel with smooth walls, where the water flow breaks instantaneously. The equations for the wave front celerity and for the dimensionless free-surface profile are given:

$$c = 2\sqrt{gh_0} \,, \tag{5.5}$$

$$\frac{X^*}{t^*} = 2 - 3\sqrt{H^*} \qquad \text{if } -1 \le \frac{X^*}{t^*} \le 2 \tag{5.6}$$

where c is the wave front celerity and the dimensionless variables are given by Equations (5.1)-(5.3). This is a frictionless solution predicting a higher speed than expected for a real fluid in contact with a rough wall.

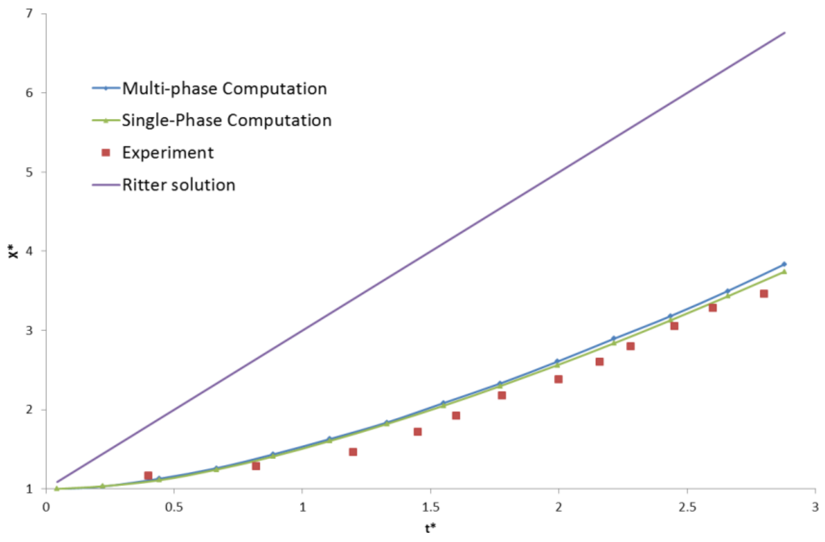


Figure 5.2: Evolution of the wave front. Comparison with experimental data and the Ritter solution for dx/h_0=0.008

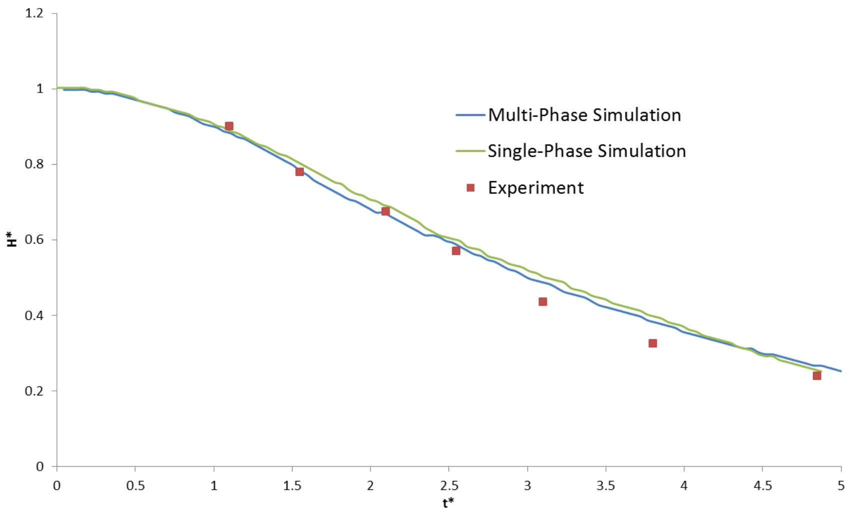


Figure 5.3: Evolution of the water column height and comparison with experimental results for dx/h_0=0.008

Figure 5.2 and Figure 5.3 show a generally good agreement for the evolution of the water flow with the experimental data, however, the flow is moving slightly faster than the experimental results. The results are replicated by the single-phase DualSPHysics code with the differences between the two simulations being exceedingly small. A difference was not expected; in the first stage of the dam break case, there is no mixing between the two phases. This result shows that the multi-phase code can reproduce a propagating water flow and has similar accuracy to the original single-phase DualSPHysics code for a flow primarily governed by the water phase.

A disparity between the experimental and the numerical results for the height of the water column appears in the latter half of the simulation, slightly before the water flow reaches the opposite wall (which happens at $t^* \cong 3$). The discrepancy is due to the dynamic boundary particles used in both codes (Crespo *et al.*, 2007) with the water particles close to the wall having a delay in their movement artificially increasing the height of the water flow. The wall effect extends to a distance ~20*h* from the boundaries.

The case has also been simulated using particle spacing dx/h_0=0.004 which corresponds to a particle number close to one million. Figure 5.4 and Figure 5.5 show the results in comparison with the experimental data and the earlier multi-phase simulation with dx/h_0=0.008. For the evolution of the water front, the behaviour is as observed in Figure 5.2; the velocity of the water flow is slightly faster than the experimental results, with the high and the low simulation exhibiting almost identical behaviour. The results are only marginally better for the high resolution.

The result for the multi-phase simulation is, at best, only marginally improved for the higher resolution even when comparing the height of the water column in Figure 5.5. This is linked to the boundary issue presented in Section 5.2.1 with the delay caused by the boundary particles being present regardless of the resolution. The results presented in Figure 5.4 and Figure 5.5, in addition to the results shown in Figure 5.2 and Figure 5.3 show that the parameters of the water flow are relatively insensitive both to the resolution and the whether the air-phase is being modelled, at least for the early stages investigated here.

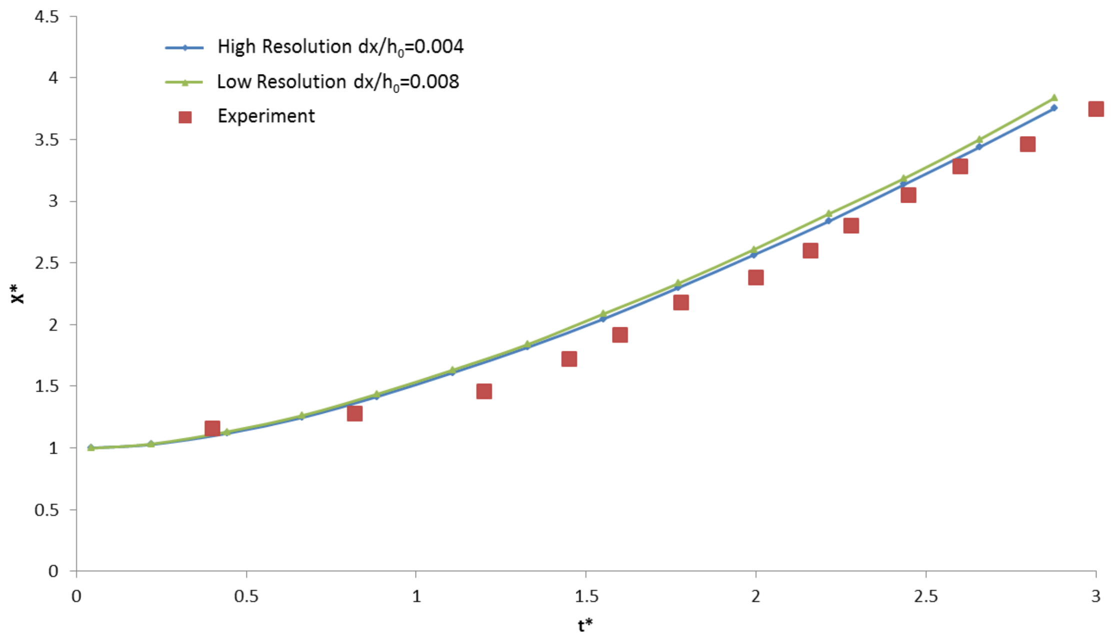


Figure 5.4:Comparison of the low and high resolution simulations for the water front evolution

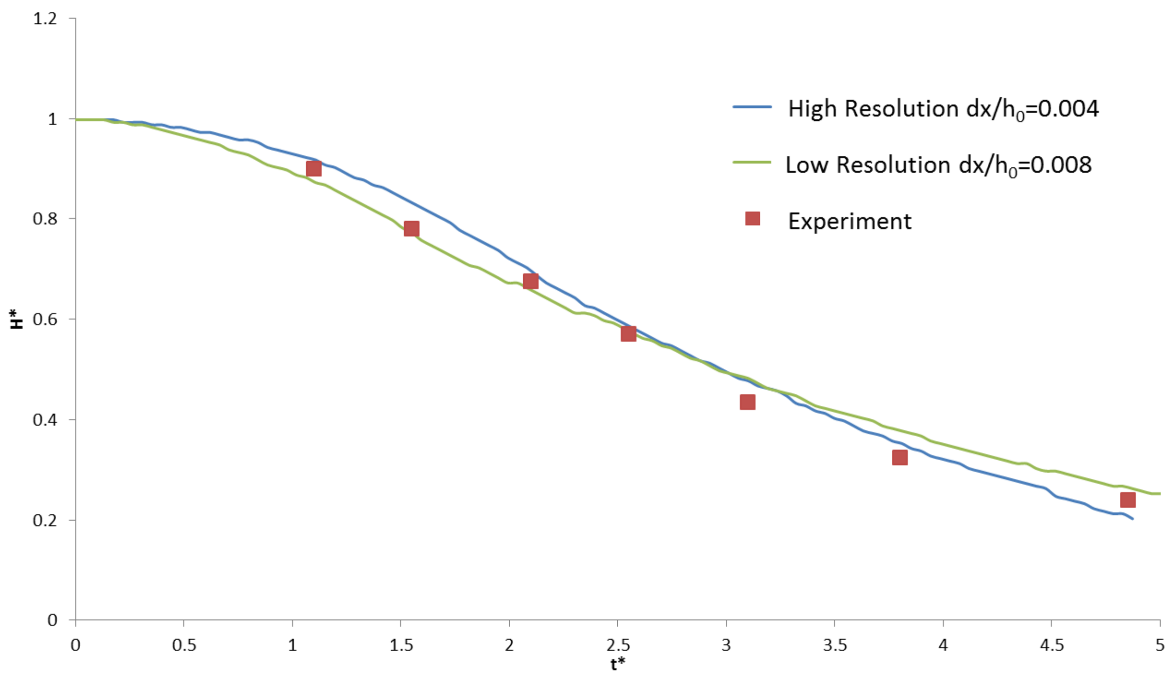


Figure 5.5: Comparison of the evolution of the water height for two different resolutions

5.3. Particle shifting strategies for weakly compressible SPH

5.3.1. Void formation in the multi-phase model

While the initial stages of the dam break test case provided close agreement for height and toe position (as compared to the experiments of Koshizuka and Oka (1996)), an issue arose when simulating large numbers of particles (over 200,000) which persists regardless of the resolution used.

Specifically, when the water reaches the opposite wall, it is eventually reflected and starts moving towards the opposite (or left-hand wall), creating a plunging wave in the process. The reflection wave is created at about 1.7s from the start of the simulation and after a further 0.1 seconds, the overturning wave shown in Figure 5.6 reaches the water at the base of the domain creating an air pocket trapped in the water flow in the process.

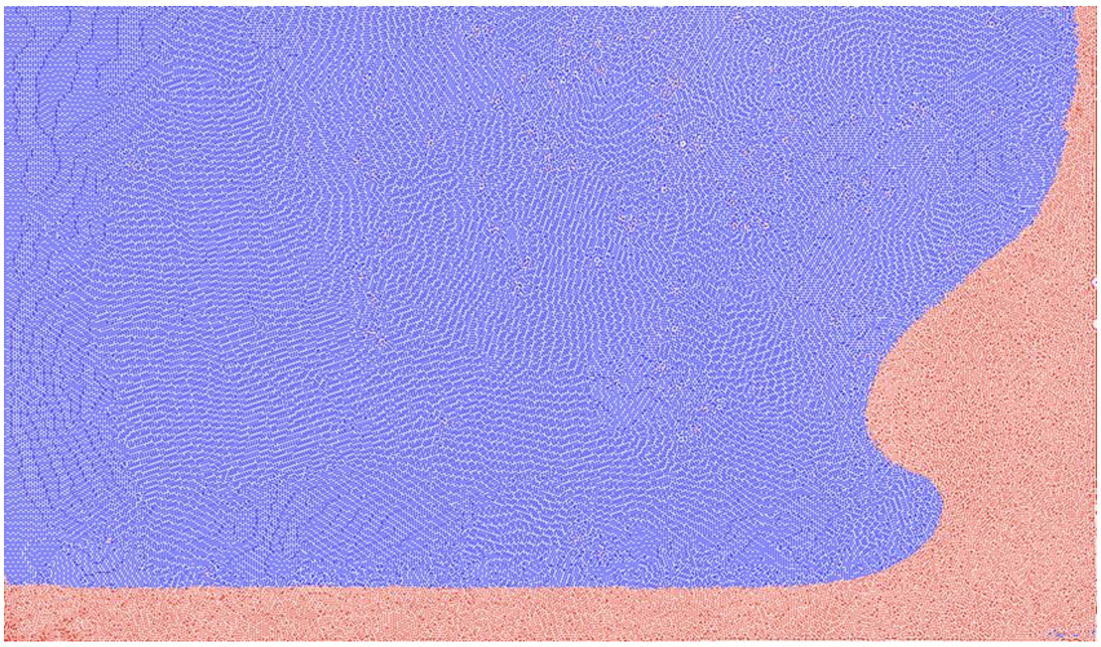


Figure 5.6: Overturning wave at 1.8s form the start of the simulation for *dx/h*$_0$=0.008

As the simulation progresses further, the air pocket is moving with the water flow; its shape constantly changing. At high resolutions however (*dx/h*$_0$=0.008), the air particles are unable to quickly adapt to the changing form and as a result voids, areas completely devoid of particles, are formed as seen in Figure 5.7, shown at 2.1s. The shape and size of the voids depends on the number of particles and they persist throughout the existence of the air pocket, except in cases where the pocket becomes small enough for the existing particles to adapt to it.

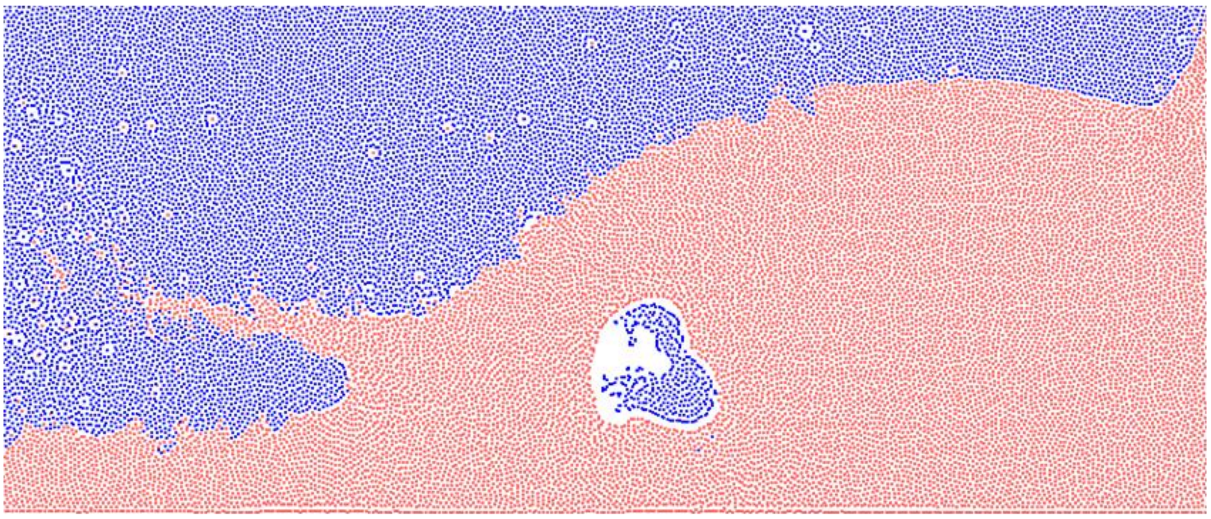


Figure 5.7: Water flow at 2.1s showing the voids being created within the water flow for *dx/h*$_0$=0.008

Apart from the voids, the interface in the air pocket also shows significant problems. The air phase is attempting to maintain its interface, creating a thin layer of particles in the areas bordering the water phase. Interaction with the water phase is as minimal as possible, as a result of a void about 2*h* that has been created in the interface separating the two phases.

The formation of the voids is due to the treatment of the air phase by the multi-phase model with the inclusion of the cohesion term, see Equations (3.58) and (3.64), the resolution or particle size and the use of single precision enforced because of the GPU. The adhesion term ensures a smooth interface between the two phases by adding a force among the air particles to prevent unphysical mixing at the interface.

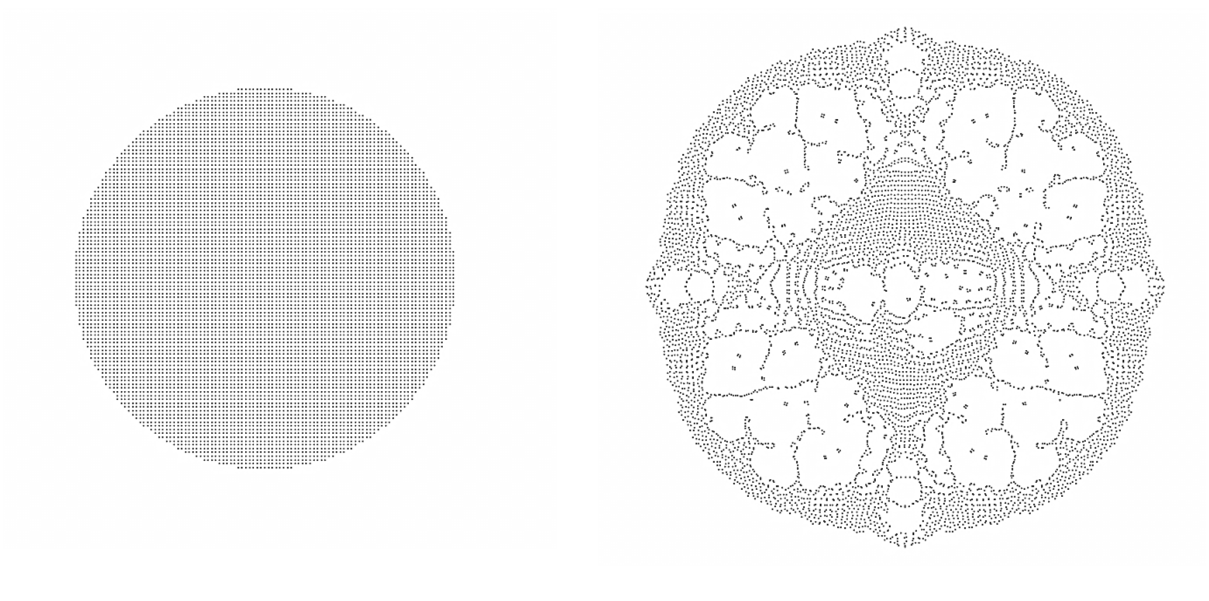
A side-effect of the cohesion force is the tendency of the air particles to remain adjacent to each other. So, when the pocket size and form change rapidly however, this force prevents the particles from separating from each other and adapting to the new shape, leading to the voids shown in Figure 5.7. In effect, the use of the cohesion term, in conjunction with the use of the same equation of state as the water phase, lead to the treatment of the air as a highly compressible liquid. Hence, the air phase cannot expand if a void is present. A different formulation such as those described in Section 2.4 might avoid this, but here we use the formulation of Colagrossi and Landrini (2003) for ease of implementation on GPU.

The voids are only present at high resolutions with initial particle spacing, dx/h_0, smaller than 0.008 and they are not dependent on the number of particles but rather on their volume and the inter-particle distance. The higher volume of the particles means that less are needed to eliminate the voids in the air pocket, which is of the same size regardless of the simulation parameters. The increased radius of the smoothing kernel at the lower resolutions also increases the particles’ movement range without breaking the free surface and explains why the issue is not observed at lower resolutions.

A secondary issue also arises as shown in Figure 5.7. The spray created by the water flow splashing against the opposite wall is now falling vertically back towards the main water body. The isolated water particles have created a zone of size 2*h* among them and the other phase so that no interaction is possible. This behaviour is initiated by the use of the Shepard filter which is computed only between particles of the same phase, but is maintained by the inability of the air phase to cover the resulting void, even though the Shepard filter is used infrequently.

In order to eliminate the voids, the empirical parameters of the Colagrossi and Landrini (2003) model were investigated and a large number of configurations tested. They did not, however, have a significant effect on the behaviour of the air particles with the voids remaining. Modifying the speed of sound for the water phase enabled slightly finer resolutions to be used, but the voids quickly re-appeared.

The behaviour of the air particles can be further examined by creating an artificial test case: a volume of air particles is placed in a domain away from the boundaries and without the effect of the gravity field as shown in Figure 5.8(a). If a constant background pressure is enforced on the air particles, the expected behaviour is a uniform expansion of the initial volume in order to cover the entire domain. This is not the case as demonstrated in Figure 5.8(b), where a constant background pressure of 100Pa was applied to 20,000 particles with particle spacing *dx*=0.01m.



a) Initial particle arrangement

b) Particle arrangement after 0.2s

Figure 5.8: Multi-phase model issue demonstrated in an artificial test case following the expansion of an air volume for *dx*=0.01m

Figure 5.8(b) shows the air particles clumping together as the initial volume is forced to expand due to the pressure gradient. The form of the volume as it expands is not random; the particles are attempting to normalise their pressure gradient (as is the expected behaviour) while maintaining a regular particle distribution and their particle concentration (number of particles in each area) gradient to 0. There is no particle diffusion being observed, which would lead the particles to areas of lower concentration. The particles instead remain in the streamlines created by the flow.

The particular arrangement in Figure 5.8(b) allows the internal particles to maintain a concentration value of 1, despite having an incomplete kernel. This is achieved only numerically by the altered particle positions. The lack of complete symmetry in the final particle position is due to the single precision of the GPU and the use of Tait's equation of state, where small density variations are translated to a substantial pressure difference and particle movement.

To eliminate the voids, the air particles must be forced to move to areas of lower particle concentration so their behaviour will approximate that of a real gas. The particles should be able to switch from one streamline to another and the particle clustering needs to be avoided. The shifting models developed by Xu *et al.* (2009) and improved by Lind *et al.* (2012b) and Skillen *et al.* (2013) are ideal for this situation.

5.3.2. Simple shifting algorithms

The shifting algorithm was initially proposed by Xu *et al.* (2009) within a divergence-free incompressible SPH approach to prevent the instability caused by anisotropic particle spacing. In this study, it has been used with the same intention, but has been slightly modified in order to apply to a weakly compressible multi-phase SPH approach instead of a fully incompressible model. This is a non-conservative approach violating momentum conservation, due to the interpolation of the hydrodynamic variables when the shifting occurs. However, this algorithm leads to smoother pressure fields and greater numerical stability in the simulation.

In the approach of Xu *et al.* (2009), after the calculation of their new positions, the particles are shifted across streamlines using the following equation:

$$\delta\mathbf{r}_s = C_{sh}\alpha_s\mathbf{R}_i \, . \tag{5.7}$$

In this equation $\delta\mathbf{r}_s$ is the shifting distance of particle i and C_{sh} is an empirical constant set from 0.01 to 1. The shifting magnitude α_s is dependent on the maximum particle velocity and the time step and is equal to their product:

$$\alpha_s = u_{\max}\Delta t \, . \tag{5.8}$$

The maximum velocity is not known before the start of the simulation but an appropriate value can be estimated with some preliminary runs. In general, the values of C_{sh} and α_s are selected so that they are sufficiently large to prevent instability but do not affect the SPH scheme by moving the particles far from their previous positions or altering the number of neighbours.

The shifting vector $\mathbf{R}_i$ is used as a weighting function to reduce the effect of more distant particles in a similar manner to the smoothing kernel. The shifting vector is given by the following equation:

$$\mathbf{R}_i = \sum_{j=1}^{M_i} \frac{\bar{r}_i^2}{r_{ij}^2} \mathbf{n}_{ij} \ , \tag{5.9}$$

where $\mathbf{n}_{ij}$ is the unit distance vector between particles i and j, M_i is the total number of neighbours surrounding the particle within the smoothing kernel and $\bar{r}_i$ is the average particle spacing among the neighbours of particle i given by:

$$\bar{r}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} r_{ij} \ . \tag{5.10}$$

After computing the shifting distance and the new particle positions, the velocity field can be updated by using a Taylor series. Xu *et al.* (2009) suggested to use the first two terms so that the order of the updating was consistent with that of the order of the Laplacian operator and be computationally efficient. Considering the velocity with a Taylor expansion:

$$u_{i'} = u_i + \delta\mathbf{r}_{ii'} \cdot \nabla u_i + O\left(\delta\mathbf{r}_{ii'}^2\right), \tag{5.11}$$

where $\delta\mathbf{r}_{ii'}$ is the distance between the shifted and the original position of the particle.

The original ISPH shifting algorithm would, at this point, have been completed, but in a weakly compressible scheme the density also needs to be updated. Initially, Equation (3.26) was used for directly computing the density through the smoothing function. The results however, were lacking accuracy, creating significant noise in the pressure field via the equation of state. To calculate the new values the Shepard filter which takes into account the density values of the neighbouring particles will be used, Equation (3.39).

A form of the shifting algorithm has already been proposed for weakly compressible SPH by Shadloo *et al.* (2011), (2012). This approach is based on the inconsistency of the particle distribution and aims to prevent particle clustering and allow a more accurate computation of the hydrodynamic gradients by enforcing a more homogeneous distribution. It has also been used with a splitting and coalescing particle scheme (Vacondio *et al.*, 2013).

5.3.3. Shifting based on particle concentration gradients

i) Previous algorithms for free-surface flow

A treatment for the air-water interface is necessary in order to use the shifting algorithm with free-surface flows. An improvement was proposed by Lind *et al.* who changed the shifting magnitude and direction according to Fick's law so that particles tend to shift towards areas with lower concentration (Lind *et al.*, 2012b), which is the expected behaviour for the air particles. In that regard using concentration gradients, Equation (5.7) was restated as:

$$\delta\mathbf{r}_s = -D\nabla C_i, \tag{5.12}$$

where *D* is a diffusion coefficient that controls the shifting magnitude and *C* is the particle concentration. Its value can be found by considering a von Neumann stability analysis of an advection-diffusion equation for concentration (Lind *et al.*, 2012b). The analysis gives the following stability condition:

$$\Delta t \leq 0.5\frac{h^2}{D}. \tag{5.13}$$

The above equation can then be used as a means of selecting the value of the diffusion coefficient. Lind *et al.* propose using the maximum value but with an additional restriction for violent flows; the particle shifting distance cannot exceed 0.2*h* to prevent significant changes in the number of neighbours as mentioned earlier.

In Equation (5.12) *C* is the particle concentration which can be found using the sum of the smoothing kernel:

$$C_i = \sum_j \frac{m_j}{\rho_j} W_{ij}. \tag{5.14}$$

The concentration gradient can be found in the same way:

$$\nabla C_i = \sum_j C_i \frac{m_j}{\rho_j} \nabla W_{ij} \,. \tag{5.15}$$

An issue that is possible to occur when using Equation (5.15) is particle clumping. Due to the dependence on the gradient of the smoothing kernel, as the distance between a pair of particles is reduced, the concentration gradient tends to zero and the particles tend to clump together (Lind *et al.*, 2012b). To correct this issue, a tensile instability correction term is used (Monaghan, 2000):

$$f_{ij} = R\left(\frac{W_{ij}}{W(\mathbf{r}_0)}\right)^n , \tag{5.16}$$

where f_{ij} is the correction term, R and n are constants introduced by Monaghan and r_0 is the initial distance among the particles at the start of the simulation. For the two constants the values suggested were R = 0.2, n = 4. Equation (5.15) can now be rewritten as:

$$\nabla C_i = \sum_j C_i \frac{m_j}{\rho_j} \left(1 + f_{ij}\right) \nabla W_{ij} \,. \tag{5.17}$$

The effects of the shifting algorithm can be demonstrated using Taylor-Green vortices. As demonstrated by Lind *et al.* (2012b), without particle shifting the particle arrangement becomes highly distorted with the particles following individual streamlines. When the shifting is introduced, a more regular particle distribution is maintained and Lind *et al.* report a variation on the particle concentration less than 2%.

An alternative approach to defining the diffusion coefficient was proposed by Skillen *et al.* (2013). They also considered the von Neumann stability analysis but imposed an additional restriction based on the particle velocity magnitude:

$$\Delta t \le \frac{h}{\|\mathbf{u}\|_i} \,. \tag{5.18}$$

Using Equations (5.13) and (5.18) and a CFL number of 0.5 the following formulation for the diffusion coefficient was reached (Skillen *et al.*, 2013):

$$D = -A_s h \|\mathbf{u}\|_i \mathit{\Delta} t \,, \tag{5.19}$$

where A_s is a parameter in the range [1,6] and $\|\mathbf{u}\|_i$ is the velocity magnitude of particle *i*. It is recommended to be set at the minimum possible value that would still allow for effective shifting minimising interpolation errors. A value of 2 is recommended.

This study uses the shifting coefficient proposed by Skillen *et al.* (2013) described in Equation (5.19) which is directly dependent on the particle velocity. The simpler term proposed by Lind *et al.* (2012b) in Equation (5.19) is not suitable in this thesis due to the very small time step enforced by the speed of sound in the air. This creates a large shifting coefficient resulting in a large number of particles exiting the domain.

The term employed by Skillen *et al.* is a more generalised form taking into account the nature of the flow. The shifting coefficient is different for each particle implicitly satisfying Fick's law, removing the need for a global shifting limit as used by Lind *et al.* Regardless of the shifting coefficient used, the algorithm still violates the momentum conservation of the particles (Skillen *et al.*, 2013), but leads to improved accuracy.

Apart from shifting the particle position, the shifting algorithm also changes the velocity of the particles in order to represent the velocity field more accurately. This was tested in this study, but the results showed that the effect, especially for high resolutions is negligible. The use of the term employed by Skillen *et al.* which regulates shifting distance according to the velocity separately for each particle instead of the global constant proposed previously is, in particular, a contributing factor to the negligibility of the velocity term of the shifting algorithm. This is in accordance with the findings of Vacondio *et al.* (2013) for weakly compressible SPH.

ii) Modified particle shifting for multi-phase weakly compressible SPH

The biggest difference between the incompressible and the weakly compressible SPH is the treatment of the density. The constantly changing density present in this study is affected by the particle position, but similar to the velocity field, changing the particle density according to the new shifted position has minimal effect on the computation. This is in accordance with the findings of Shadloo *et al.* (2011) and Vacondio *et al.* (2012), neither of whom modify the density.

The iterations of the algorithm for the incompressible SPH utilised the mirror particle approach while the present study uses the dynamic particles (Crespo *et al.*, 2007). Following the previous approaches the boundary particles were initially not participating in the shifting computation, with an explicit limit enforced on the fluid particles so that exiting the domain was not possible. However, this leads to particles clumping at the interface as they were attempting to move to the lower concentration areas and eventually leaving the system due to the force exerted from the other particles. Hence, the boundary particles will be used when calculating the concentration and the concentration gradient of the particles to ensure that the fluid particles near the edge of the domain have a complete kernel and prevent the particles from moving in that direction. For the air particles, the boundaries may have different density, but both the concentration and its gradient calculated use the particle volume, which is of the same order of magnitude for each particle. Hence, Equation (5.15) will be modified as follows:

$$\nabla C_i = \sum_{j \in F \cup B} C_i \frac{m_j}{\rho_j} \nabla W_{ij}, \tag{5.20}$$

where *F* and *B* denote the set of fluid and boundary particles respectively.

Further consideration must be given to the tensile instability term (Monaghan, 2000). It is only necessary for spline kernels, being used as a supplement to the shifting algorithm in order to compensate for the discontinuities of the quintic spline kernel used in the incompressible SPH algorithm by Lind *et al.* (2012b). The present study however, uses the quintic Wendland kernel so this correction will not be used.

This is not the only difference between this study and the investigations of Lind *et al.* (2012b). In the incompressible SPH model the quintic spline kernel used has been normalised in order to achieve higher accuracy (Oger *et al.*, 2007). The normalisation leads to a different form of the equations which calculate the concentration gradient (Equations (5.15) and (5.17) omitting the particle concentration. In this study, the normalised kernel is not used due to restricted functionality of the GPU code; therefore Equation (5.20) will be solved using the normal kernel gradient.

The benefit gained by the application of the shifting algorithm can be demonstrated by the artificial test case shown in Figure 5.8. Figure 5.9 shows the particle positions using Equation (5.12) for particle shifting, Equation (5.19) for the calculation of the diffusion coefficient and Equation (5.20) for the calculation of the concentration gradient. The particle distribution is much smoother and the concentration is consistent. The placement of the boundaries near the free surface however, is not symmetrical due to the incomplete kernel and the single precision of the GPU.

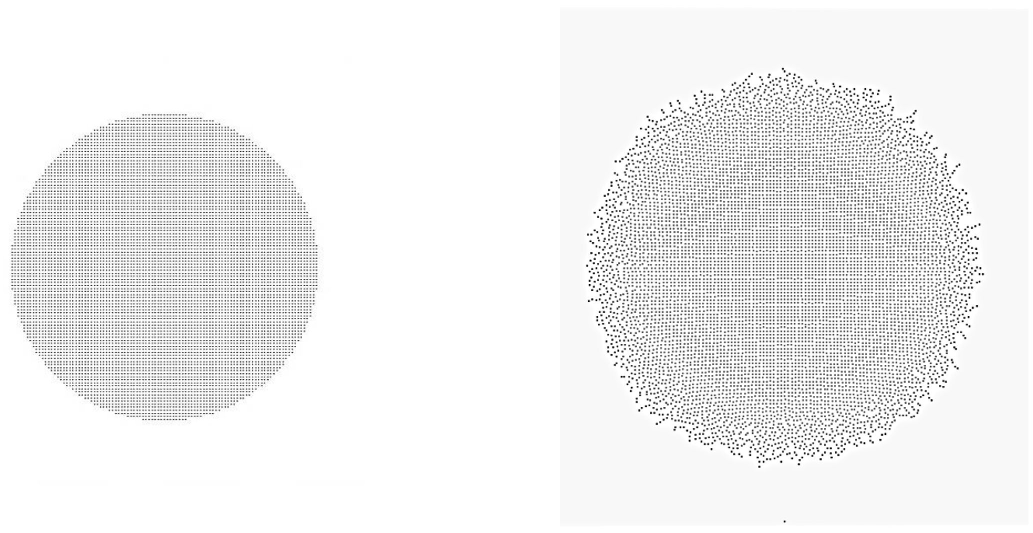

a) Initial particle arrangement b) Particle arrangement after 0.2s

Figure 5.9: Expansion of an air volume after the application of the shifting algorithm

5.3.4. Modification of the Free-surface Correction

Apart from the improvements in determining the shifting distance and direction, Lind *et al.* proposed a correction for the free surface. The use of Equation (5.12) results in a constant movement of the particles towards the free surface due to the large concentration gradients present at the interface. This was also confirmed by the tests made in this study. Despite using a multi-phase model that modelled the air so that the test cases examined do not have any surface devoid of particles, there were severe issues at the interface between water and air particles, with the lighter phase dispersing in the heavier one.

In addition, instabilities were found in violent flows with the interface breaking and the appearance of voids. To prevent the unphysical movement at the free surface the concentration gradient near the surface is not controlled using the global coordinates but rather the local tangent and normal vectors at the free surface:

$$\delta\mathbf{r}_s = -D\left(\frac{\partial C_i}{\partial s}\mathbf{s} + \alpha_n\left(\frac{\partial C_i}{\partial n} - \beta_n\right)\mathbf{n}\right), \tag{5.21}$$

where ***s*** and ***n*** are the tangent and normal vectors to the free surface, while β_n is a reference concentration gradient in a still free surface. The parameter α_n limits the diffusion at the normal direction of the free surface; for violent flows α is set equal to 0. However, for long slow flows errors due to incompleteness of the kernel can potentially accumulate at the free surface (Lind *et al.*, 2012b). A small degree of diffusion at the normal direction is then allowed with the parameter α set equal to 0.1.

To identify the free surface the method of Lee *et al.* (2008) is used with the divergence of the particle position being computed:

$$\nabla \cdot \mathbf{r} = \sum_j \frac{m_j}{\rho_j} \mathbf{r}_{ij} \cdot \nabla_i W_{ij} \,. \tag{5.22}$$

The value of the divergence for a full kernel is equal to 2 for a 2-D case and 3 for a 3-D case; for a particle at the free surface is naturally much less. The value of 1.5 is proposed as a threshold value for a 2-D computation to locate the surface particles. This method does not offer absolute precision, it is possible for some free surface particles to be ignored (Lee *et al.*, 2008), the error however appears to be small (Lind *et al.*, 2012b).

Equation (5.21) is only suitable for a two-dimensional simulation. For the 3-D case, the binormal vector also needs to be taken into consideration. The aim of the surface term is to prevent the free surface from breaking by restricting shifting in the normal direction which is orthogonal to the free surface. Following on that, the 3-D implementation shown in Equation (5.23) should allow for shifting only on the tangent and binormal directions, treating the normal direction in the same way as the two-dimensional implementation:

$$\delta \mathbf{r}_s = -D\left(\frac{\partial C_i}{\partial s}\mathbf{s} + \frac{\partial C_i}{\partial b}\mathbf{b} + \alpha_n \left(\frac{\partial C_i}{\partial n} - \beta_n \right)\mathbf{n} \right), \tag{5.23}$$

where **b** is a unit vector orthogonal to **s** and **n**. Similarly, the threshold value of $\nabla \cdot \mathbf{r}$ computed using Equation (5.23) for 3-D simulations is taken as 2.5 as used by Muhammad *et al.* (2013).

The free surface is a very important issue for an air-water multi-phase model. The extra terms introduced in the multi-phase model prevent, as mentioned, the air phase dispersing on the water phase (Colagrossi and Landrini, 2003). The shifting algorithm has the potential to severely disrupt the interface, especially considering that the initial algorithm devised by Xu *et al.* (2009) had significant issues with the free surface (Lind *et al.*, 2012b) creating unphysical particle movement in the free surface.

The surface term should, in principle, preserve a smooth interface although there is the possibility that the voids will not be adequately treated. The free-surface correction in Equation (5.23) may further enhance the cohesion term for the air particles at the interface preventing their movement and negating the advantages of the shifting algorithm. To be effective, the shifting algorithm needs to allow the air particles to expand without displacing the water phase, while having minimal impact on the latter.

A test will demonstrate the application of the surface term on the multi-phase model. Initially, the dam break cases are run using the surface term in both phases, but the term is applied to either only the water phase or neither of them. To achieve minimal interference of the shifting algorithm with the water flow, a dam break case where the algorithm only applies to the air phase will be run. For the dam break test case the second term in Equation (5.21) is not needed, the flow has sufficient speed and a rapidly-evolving interface for the term to be significant (Lind *et al.*, 2012b) so the value of the parameter *a* has been set to 0.

A) Using the surface correction term for both phases

The surface correction was initially used for both phases to ensure a smooth free surface. The smaller voids surrounding the isolated water particles are a lesser factor but when the air is trapped in the pocket, there is no particular difference when compared to the simulation without shifting, as observed in Figure 5.10, which is the same time instant as Figure 5.7. A separation at the interface below the plunging wave is also noticeable.

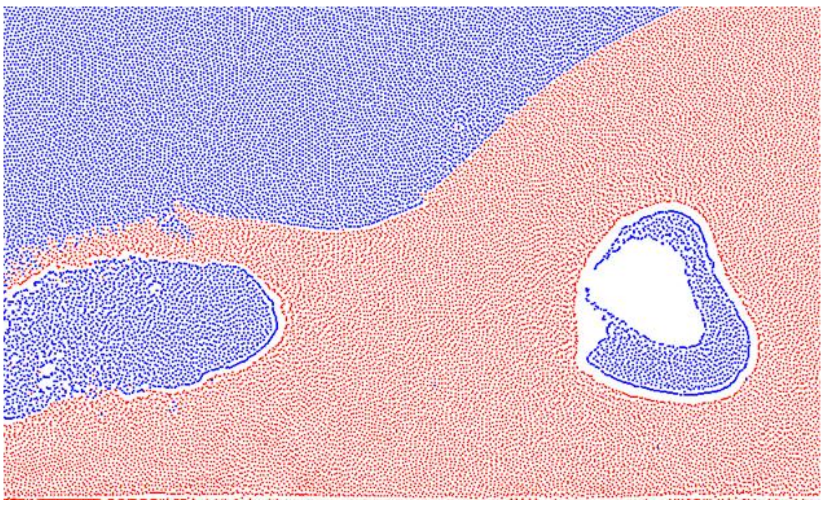


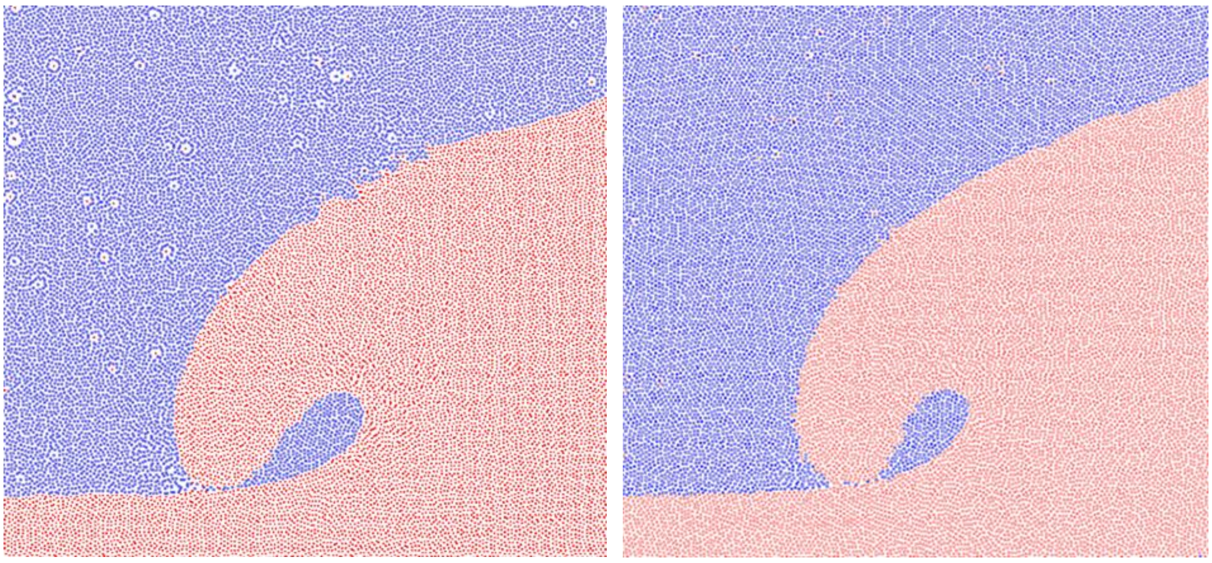
Figure 5.10: Dam break flow at 2.1s after using the surface correction term of Equation (5.21) for both phases for dx/h_0=0.008

Looking at the interface and at the air pocket, there is a thin layer of particles in each phase, forming the free surface. Particles in that layer are clustered, especially in the air phase, preventing any particle from moving to an area of low concentration. This is due to the effect of the surface term; removing shifting for the normal direction prevents the particles from changing the free surface.

The air phase cannot expand properly; the surface term has a similar effect to applying surface tension, causing it to behave entirely like a liquid and making the treatment of the voids impossible. The effect is however, advantageous for the water phase as the free surface is not interrupted by the shifting algorithm.

B) Using the surface term only for the water phase

When using the surface term, the dam break case does not present many differences from the simulation without shifting at the initial stages of the flow. However, at the point where the overturning reflected wave touches down, differences between the two simulations can be discerned. In Figure 5.11, the air pocket created by the shifting simulation is slightly smaller and without shifting, a small void is created at the intersecting point of the overturning wave to the main flow. Small ripples have also begun to appear in the water surface as the spray created by the water crashing violently on the opposite wall return to the main flow.

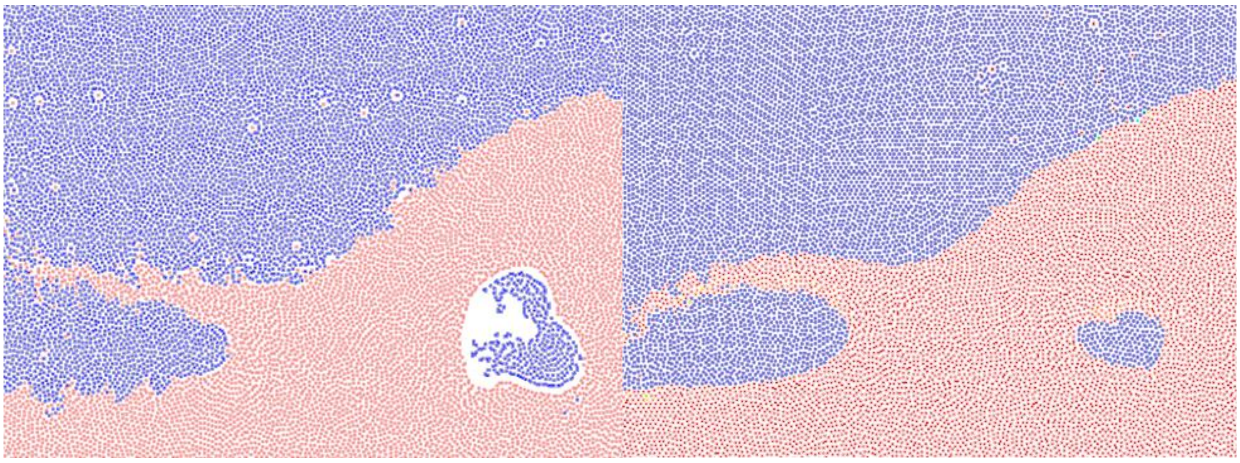

a) Without shifting in either phase

b) With shifting in both phases and the surface correction term only in the water phase

Figure 5.11: Comparison of the reflected water flow at 1.95s for *dx/h*$_0$=0.008

The most significant change on the behaviour of the air particles is observed when they are trapped within the water flow. Figure 5.12 presents a comparison between the original simulation and the shifting algorithm without the surface correction. The removal of the surface term has allowed the air particles to freely move to areas of lower concentration and as a result no voids are occurring in the air pocket, regardless of the constant change in its shape. Compared to the computation using the surface term for the air particles in Figure 5.10 this result is much closer to the expected air behaviour.

Fluid particles are more evenly spaced regardless of the phase of the neighbouring particles. The interface is also much smoother in Figure 5.12b, while the suspended water particles are causing disturbances as they return in the main flow in Figure 5.12a. This is most prevalent in areas with high velocities. The resulting air pocket in Figure 5.12b has a smaller volume than the original computation in Figure 5.12a and the particles, especially the water particles in the interface, are more evenly spaced. The interface, following from Figure 5.12 is smoother and better defined, with the suspended particles having minimal effect even when the number of water particles is lower.


a) Without shifting in either phase

b) With shifting in both phases and the surface term only in the water phase

Figure 5.12: Comparison of the reflected water flow at 2.1s

The inclusion of the shifting algorithm has significant effects on the pressure field. As seen in Figure 5.13(a) small differences in the pressure field can be seen even at 1s, Figure 5.13(b) shows that the pressure contours are more homogeneously defined in the area close to the interface if the new shifting algorithm is used. The flow displayed in Figure 5.13(a) at *t*=1s shows some pressure fluctuations, especially where the water is rising as the number of water particles in that area is significantly smaller. In contrast, using the shifting algorithm restricts these fluctuations at the highest point of the flow.

More differences can be seen in Figure 5.14, which depicts the simulations at 2s, shortly after the air entrainment. Small voids begin to appear in Figure 5.14a, and the pressure around the bubble shows a significant and sudden decrease. A small decrease in pressure around the bubble can be also seen in Figure 5.14b, but it is much smaller and smoother. In Figure 5.14a, of note are the discontinuities that appear in the left side of the domain, at the starting point of the water column and a general decrease in pressure close to the boundaries which does not appear in Figure 5.14b.

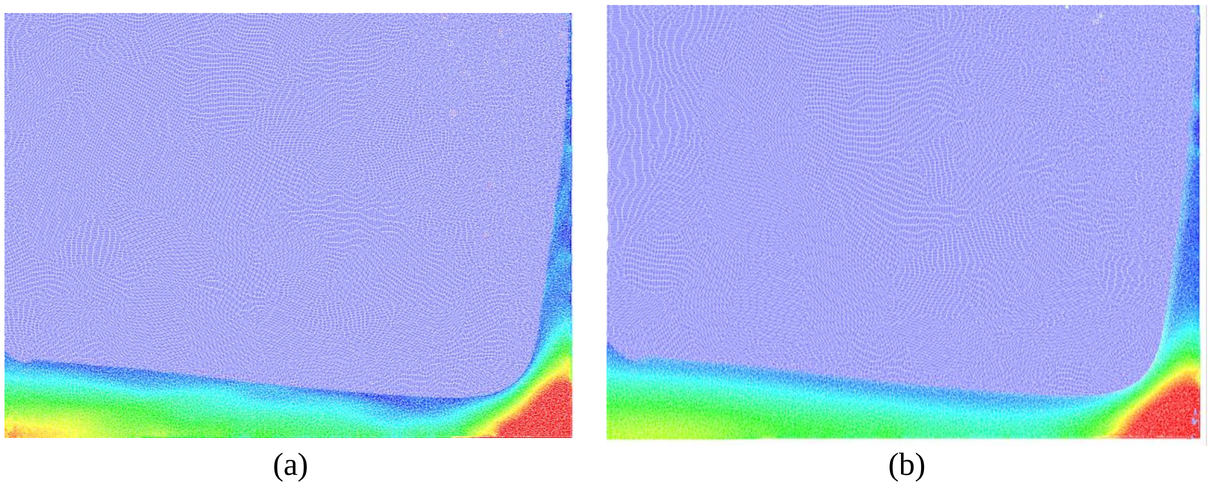


Figure 5.13: Comparison of the water pressure field at 1s for two simulations a) without the shifting algorithm b) with the shifting algorithm

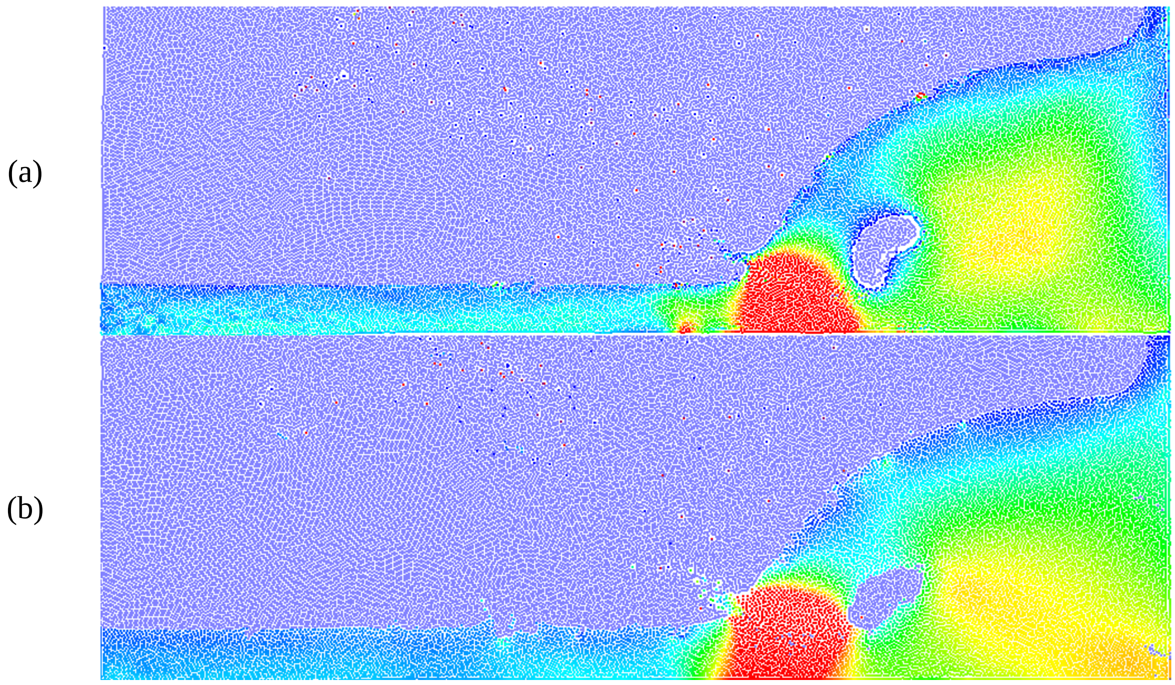
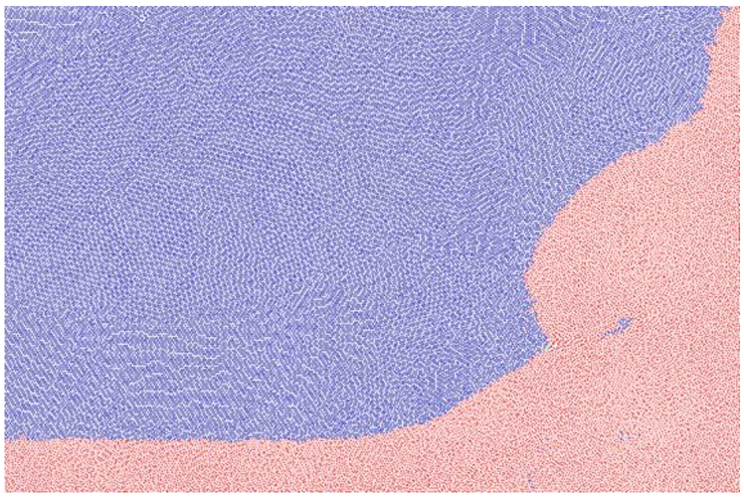


Figure 5.14: Comparison of the water pressure field at 2s for two simulations a) without the shifting algorithm b) with the shifting algorithm

C) Removing the surface correction term for both phases

Examining further the role of the surface term on the shifting algorithm, a simulation using shifting particle positions for both phases, but without the surface term, was executed. The results, shown in Figure 5.15 produced a different profile for the water phase. Its behaviour after it is reflected from the contact with the right wall is not consistent with the expected dam break results (Colagrossi and Landrini, 2003). Specifically, the toe of the reflected wave collapses almost immediately on the main flow, resulting in a very small air pocket being created when compared to Figure 5.11. The resulting splash occurs then much earlier that it is supposed to, with the expected behaviour seen in Figure 5.15, trapping a large air volume in the process.

a) Flow at 1.85s



b) Flow at 2.35s

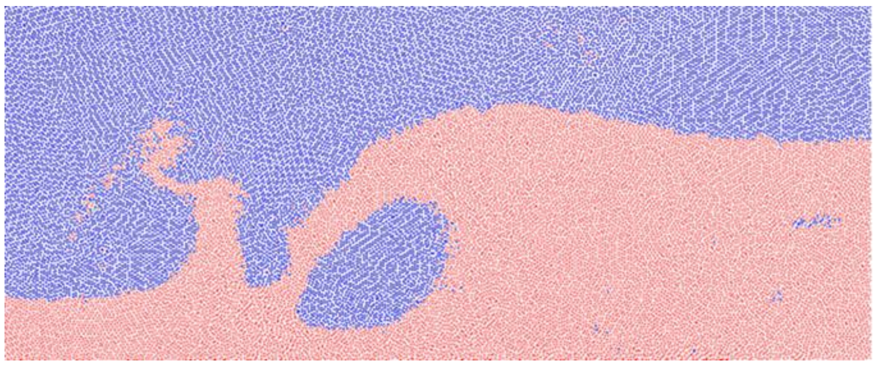


Figure 5.15: Water flow at different instants for simulation with shifting but without free-surface correction in either phase

The reason for this behaviour is the large shifting distance permitted for the water particles. The movement of the water flow due to gravity results in large forces being exerted on the air particles as a result of the greater mass of the water particles. The resulting movement of the air phase creates zones of low concentration. In the previous shifting simulations, the air particles would then move towards their former positions, minimising the resulting concentration gradient.

Without the restriction of the free surface correction, however, the water particles are now free to move to these zones similar to the air particles. This behaviour results in a further displacement of the air particles in the next time step due to the new forces exerted, eventually resulting in the profiles seen in Figure 5.15. In essence, the shifting algorithm exacerbates the existing water movement due to the gravity force beyond what is physically possible, while the air particles provide minimal resistance. This behaviour is also encountered if the surface term is only applied to the air phase.

D) Use of the shifting algorithm only for the air phase

The use of the surface correction is then essential in restricting the water movement when the shifting algorithm is used. An additional possibility however, is to avoid the use of the shifting algorithm entirely when computing the water phase. The profile for the water phase produced by the original multi-phase model in Figure 5.12a is similar to the profile given in Figure 5.12b and shifting the particle positions for the air phase should avoid the creation of voids or the disturbances at the interface.

Indeed, the results presented in Figure 5.16 show that the majority of these errors are addressed even without shifting the water particles. Voids are not created; neither within the air pocket, nor within the main air flow and the interface remains relatively smooth. The only issue occurs in the secondary splash-up caused by the plunging water wave. Due to the effect of the spray it is not as well defined and there is significant mixing between the two phases.

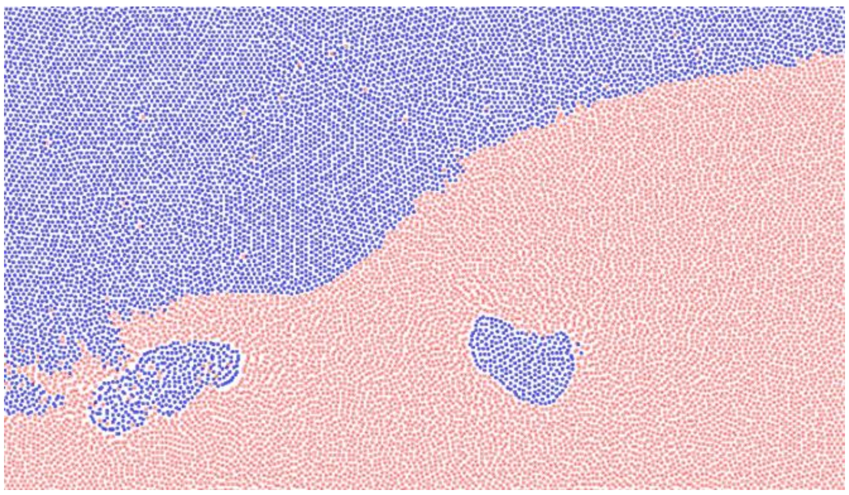


Figure 5.16: Water flow at 2.1s with the shifting algorithm used only for the air phase

The splash-up continues to be less defined as the simulation progresses. Figure 5.17 shows the evolution of the splash-up. The use of the shifting algorithm allows for the free surface to remain undisturbed, while without shifting the forces exerted by the air phase disperse the splash-up due to the small number of water particles present.

a) Shifting algorithm only used for the air particles

b) Shifting in both phases with the surface term applied only in the water phase

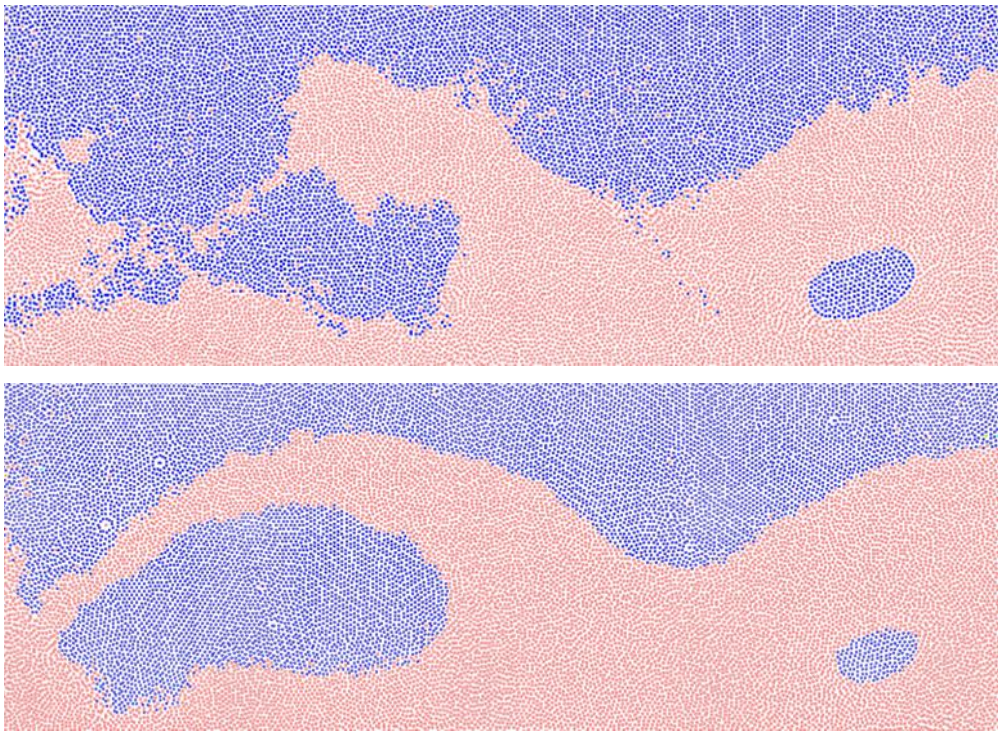


Figure 5.17:Comparison of the water flow at 2.5s

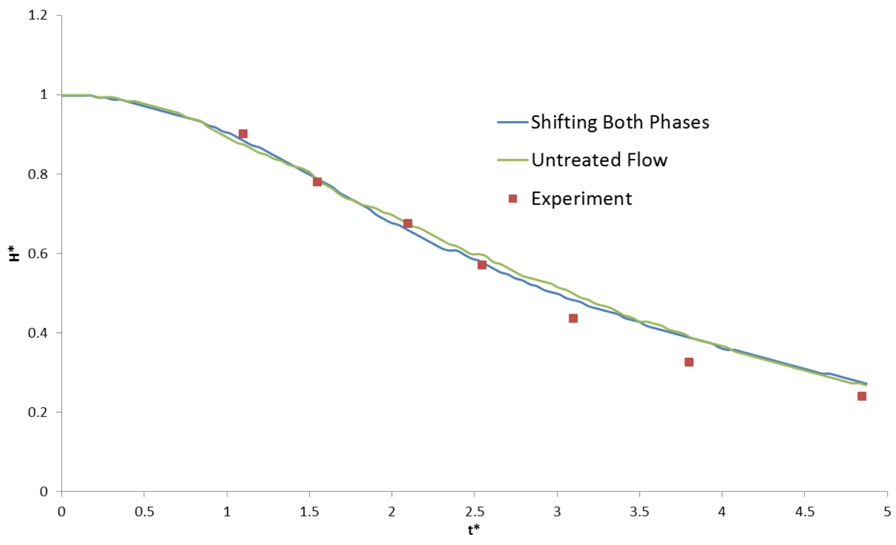
5.3.5. Shifting validation

To determine whether the multi-phase simulation still gives the correct results even after using the shifting algorithm and whether the algorithm needs to be used in the water phase, the dam break test case described in Section 5.3 will be used.

i) Height and toe position

The results presented in Figure 5.18 show the height of the water column as it collapses; they are presented for two simulations both of which use the shifting algorithm in the air phase. The water phase however, is treated only for one of the two simulations. The results are also compared with the experimental data of Koshizuka and Oka (1996) and the multi-phase simulation without any shifting treatment, presented in Section 5.3.2.

(a)



(b)

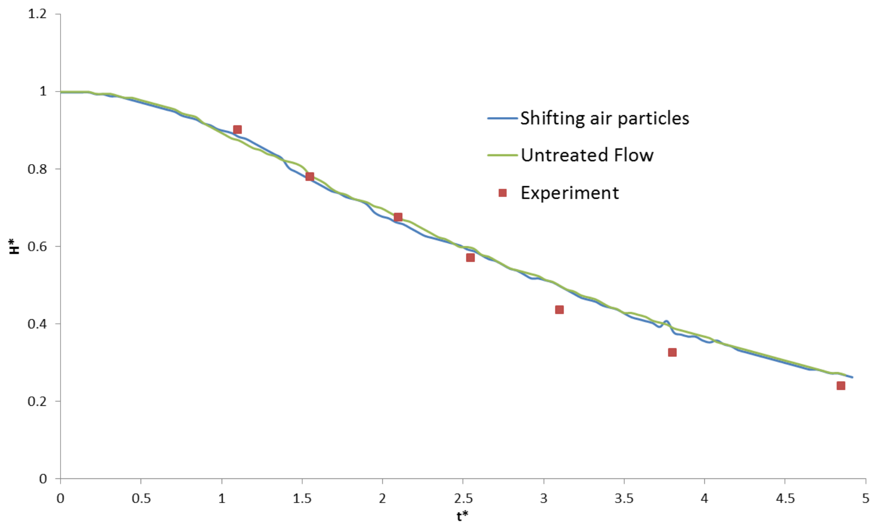


Figure 5.18:Comparison of the water column height for a)a simulation with both the air and the water phase shifted and b) a simulation with only the air being shifted for *dx/h_0*=0.008

As seen in Figure 5.18, differences between the multi-phase simulations are minimal. When using the shifting algorithm, there is a very slight deviation in the middle, but in general the shifting algorithm has no significant effect in the water flow. This is an expected result for the dry dam break case, since the water movement and velocity is dictated by the gravity forces, while the air-water interaction has a lesser effect, especially in the early stage of the flow as seen in Section 5.3.2 where the multi-phase and single-phase results displayed no significant differences as well. It also shows however, that the shifting algorithm cannot correct the error introduced by the fictitious boundary particles.

Apart from the height of the water column, the toe position of the water flow can also be compared with experimental data. The results are similar to the ones presented in Figure 5.2 and Figure 5.4 with very slight deviations only appearing when using the shifting algorithm in the water. It can then be concluded that in the dry dam break flow for the water phase, shifting the air or the water particles does not significantly alter the flow, introducing errors or improving the results.

ii) Effects of shifting on pressure field

The benefits gained from using the shifting algorithm are more visible in the later stages of the flow especially when the overturning wave and the resulting splash-up are created as seen in Section 5.4.4. However, the experiment by Koshizuka and Oka (1996) does not have information for the later stages of the flow. In that regard, a different dry dam break case has been executed. This case was first simulated by Colagrossi and Landrini (2003) using their multi-phase model and an incompressible boundary element method (BEM) by Faltinsen *et al.*(2004) and includes numerical results regarding the overturning wave. The case has also been simulated by Lind *et al.* (2012a).

The basic elements of the flow as well as the driving forces behind the flow are the same as the dry dam break by Koshizuka and Oka (1996). The difference lies in the dimensions of the water column and the tank, specifically the distance between the two walls. The definition sketch is shown in Figure 5.19, where the dimensions are dependent on the height of the water flow h_0. Colagrossi and Landrini (2003) set the geometric parameters for this case as l_0/h_0=2, H/h_0=3 and L/h_0=5.366.

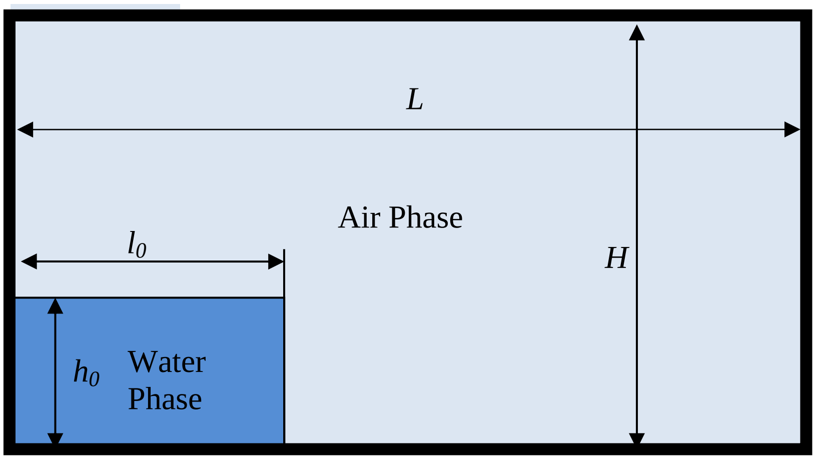


Figure 5.19:Definition sketch for the second dry dam break case

The other parameters of the flow are also provided by Colagrossi and Landrini (2003) as a function of the water column height. In their simulation they used 39072 fluid particles, 4900 of which were used for the water flow resulting in particle spacing dx/h_0=0.002. The speed of sound used for this case is $c_{s,w} / \sqrt{gh_0} = 10.9$ and $c_{s,a} / \sqrt{gh_0} = 155$ while the smoothing length and the time step are set as $h / h_0 = 2.69 \times 10^{-2}$ and $\Delta t \sqrt{gh_0} = 4.51 \times 10^{-4}$ respectively.

The simulation was tested for different configurations, each one using different values of the density re-initialisation term and the artificial viscosity. It is important to note that these terms are not the same as the current study. Colagrossi and Landrini (2003) used the density re-initialisation of an MLS filter as described in Section 3.3.2 and for the artificial viscosity the term proposed by Monaghan and Pongracic (1985) was used but modified according to the work of Balsara (1995).

To identify the effect of shifting on the pressure fields and compare it with a reference solution, various combinations of shifting density filtering and viscosity are investigated. The results for the different configurations are first presented for an instance at $t\sqrt{g / h_0} = 2.98$. In this particular study, the height of the water column H was selected as 1m so the corresponding time is 0.95s. The instants are before and after the impact of the water front against the vertical wall at the right side of the domain. The results are presented in Figure 5.20 and Figure 5.21.

The results presented are (a) from the simulation by Colagrossi and Landrini (2003) and three multi-phase simulations with different applications of the shifting algorithm: in simulation (b) the flow is untreated, simulation (c) uses the shifting algorithm only for the air phase, while in simulation (d) both phases are shifted. Simulation (e) shows the corresponding results of a single-phase simulation using the original DualSPHysics code. Simulations (a)-(d) do not show the air phase for clarity.

A) No density filter

Removing the density re-initialisation from the simulation has an apparent effect on the pressure field as seen in the first column of Figure 5.20. While the pressure contours are maintained close to the initial position of the water column, the toe of the flow shows a noisy field, especially close to the boundaries. The results using the DualSPHysics code, with or without the multi-phase model and the shifting algorithm are however, less noisy than the results of Colagrossi and Landrini (2003). The particle spray is also not observed for the simulation (a); it occurs in every other instance. The reason for this discrepancy lays in the boundary particles, while simulation (a) uses a mirror particle approach, simulations (b)-(e) use the fictitious boundaries approach by Crespo *et al.* (2007). Due to the high pressures and forces generated by the impact the noise is also worse at the right side of the domain.

B) No viscosity model and no density filter

Removing both the viscosity model and the density re-initialisation term results in a noisy pressure field with no distinct pressure contours as seen in the second column of Figure 5.20. Areas that have a smaller number of water particles or are near the free-surface tend to be more affected from this noise. The results are quite similar between every simulation, with only the single-phase showing a pressure field with lower values. The removal of the viscosity model increases the pressure noise generated, especially in areas with a sufficient number of particles. The spray is again only observed for the DualSPHysics simulations.

C) No viscosity model

The first column in Figure 5.21 shows the results for a simulation which does not use any viscosity model. For $t\sqrt{g/h_0} = 2.98$ the pressure contours are well represented without many differences between the simulations, although simulations (a) and (e) show a lower pressure field compared to the others. The issue with the boundaries moving the fluid particles upwards is also observed here; it is especially prevalent in simulation (e). Of note are the differences in the rising water: the Colagrossi and Landrini (2003) simulation has progressed further but no spray has been created. For the single-phase model the rising water is pushed away from the wall under the effect of the boundary particles.

D) Use of both viscosity model and density filter

The results obtained when using a density filtering term are similar as shown in the second column of Figure 5.21, regardless of the particle viscosity. Results for this case by Colagrossi and Landrini (2003) are not available. The pressure contours are well-defined and the field is relatively pressure-free. Use of the shifting algorithm does not create significant differences; few fluctuations in the pressure filed can only be seen in simulation (c). The behaviour of the rising water is the same as observed in the previous simulations with a water spray being created and the particles being pushed away from the wall.

For the resolution and the parameters used in this case, the shifting algorithm does not have any immediate effects on the results obtained. The pressure is instead greatly dependent on the density filtering to maintain a noise-free filed. The evolution of the flow does not show any large changes, but the choice of boundary conditions does affect the spray and the rising flow on the right-hand wall after the impact.

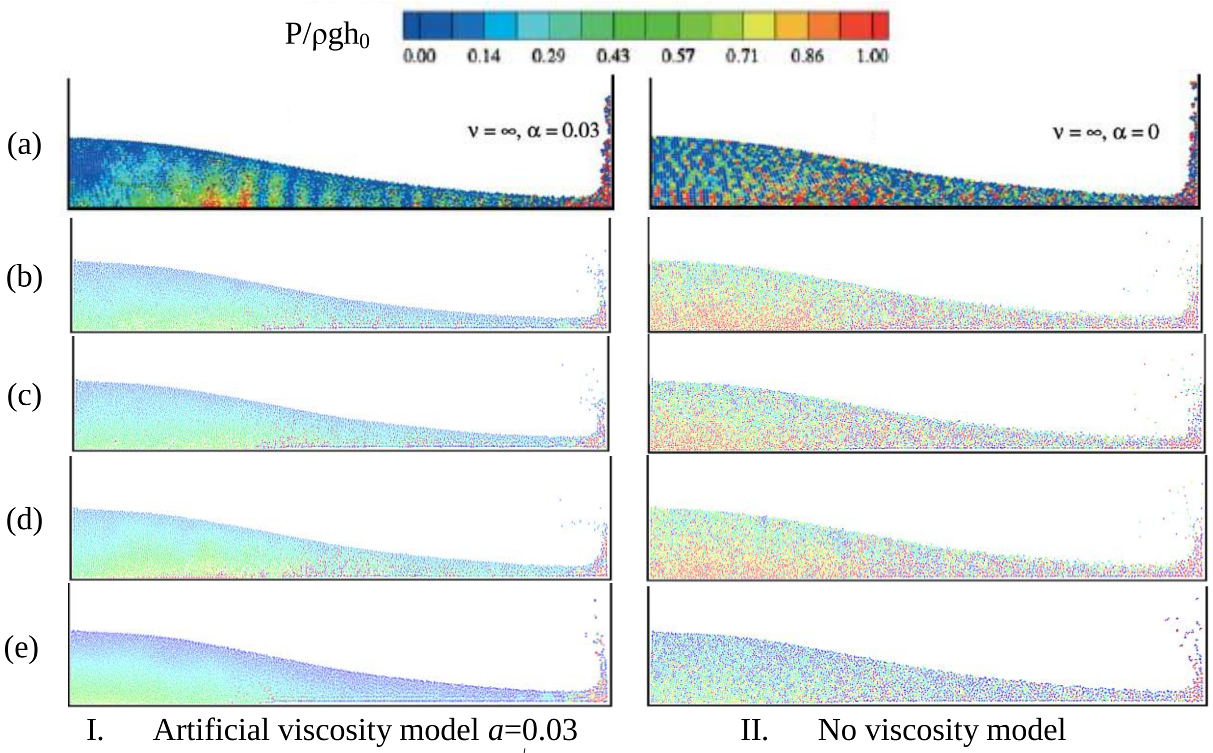


Figure 5.20:Comparison of the pressure field at $t\sqrt{(g/h_0)}$=2.98 for dx/h_0=0.002 without density re-initialisation for a) Colagrossi and Landrini (2003) b) present model without shifting c) present model with shifting only in the air phase d) shifting in both phases e) single-phase simulation. Simulations (a)-(d) do not display the air phase for clarity

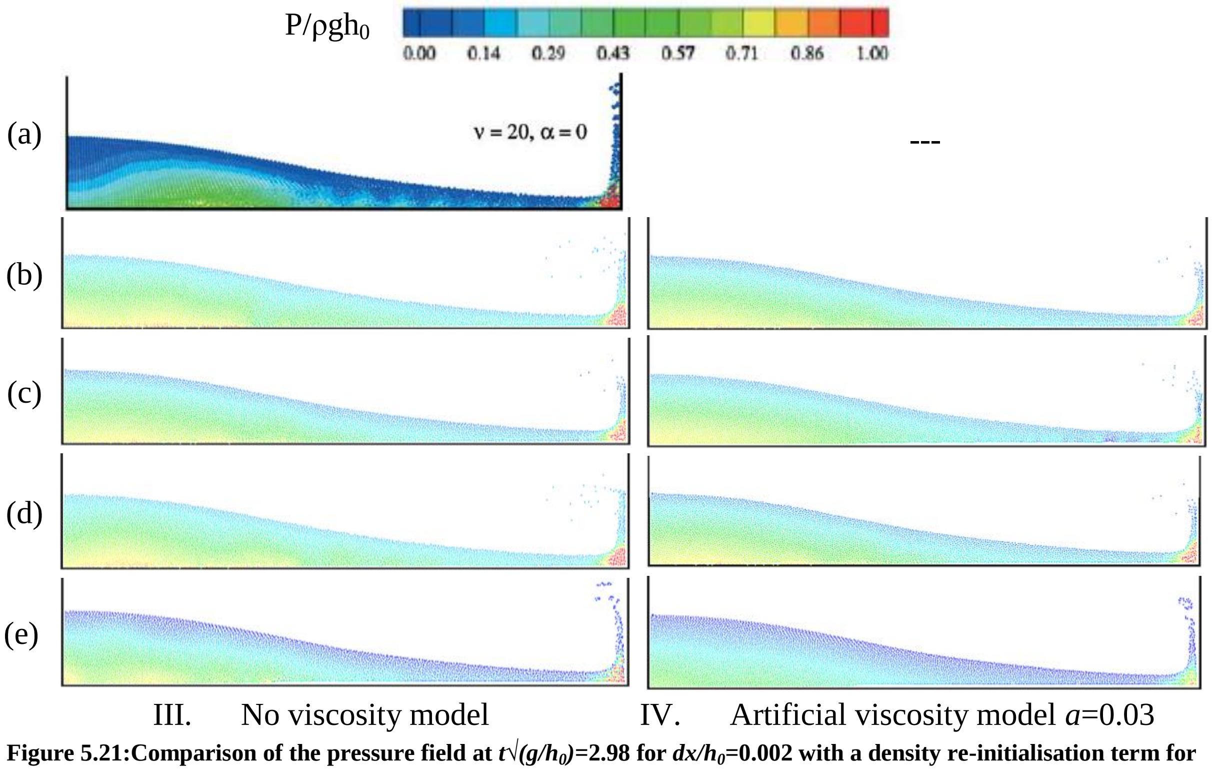


Figure 5.21:Comparison of the pressure field at $t\sqrt{(g/h_0)}$=2.98 for dx/h_0=0.002 with a density re-initialisation term for a) Colagrossi and Landrini (2003) b) present model without shifting c) present model with shifting only in the air phase d) shifting in both phases e) single-phase simulation. Simulations (a)-(d) do not display the air phase for clarity and results for simulation (a) are not available for situation IV

Figure 5.22 shows the result for the same test case but with the particle distance halved increasing the particle number to 160,000. The new smoothing length is then $h/h_0 = 1.3\times10^{-2}$ and the particle spacing is dx/h_0=0.001. The parameters for the equation of state remain the same as in Colagrossi and Landrini (2003) and the previous simulations. The results are presented for two values of the artificial viscosity coefficient, 0.03 and 0. The results use the Shepard filter every 20 time steps to maintain a noise-free pressure field.

Figure 5.22 shows the results at $t\sqrt{g/h_0} = 2.98$ for the same viscosity coefficient. Very few differences can be seen in the shape of the flow, with only the rising water flow being slightly higher and thinner with the viscosity, while in the other case the particles have a larger spread and its tip is less well defined. Compared to the dry dam break case presented in Section 5.2.2 there is no delay in the movement of the particles close to the wall but the effect of the boundary particles can be seen in the pressure field.

Bigger differences however, can be seen in the pressure field. The pressure for simulation (c) where the shifting algorithm is used in both phases is overall increased. The effect is more apparent on the free surface in Figure 5.22 without the artificial viscosity and indicates a similar increase in the air pressure. With the viscosity model on the other hand, a smaller increase in the free surface is observed, although it is still noticeable when compared to simulations (a) and (b), with a higher increase in the left side of the domain at the starting point of the flow. Pressure results for the rising water flow at the opposite wall remain unaffected.

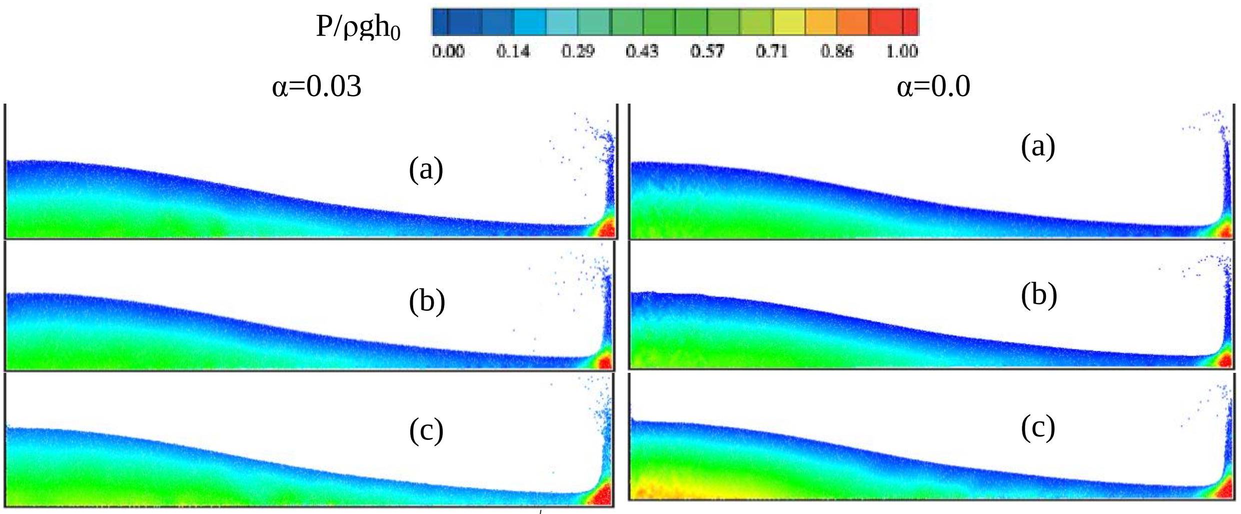


Figure 5.22: Comparison of the pressure field at $t\sqrt{(g/h_0)}$=2.98 for dx/h_0=0.001 without a viscosity model for a) present model without shifting b) present model with shifting only in the air phase c) shifting in both phases. Simulations do not display the air phase for clarity

Compared to the pressure field for the lower resolution, shown in Figure 5.20 and Figure 5.21, differences exist in both the pressure field and the particle position. Specifically, when considering the particle position, the height of the rising flow is increased for the higher resolution. This is due to the lower numerical viscosity for this simulation and it occurs regardless of the use of the shifting algorithm or the value of the artificial viscosity coefficient.

The other difference is on the pressure field, where the lower resolution cases show increased pressure for the water particles, similar to the simulation with shifting in both phases for the higher resolution. The increased pressure seems then to be linked to three parameters: the particle resolution and the shifting algorithm, while the artificial viscosity value affects both its final values and its distribution.

iii) Overturning Wave Free-Surface Comparison

A different set of comparisons examines the SPH simulation in comparison with an incompressible Boundary-Element Method (BEM) from Faltinsen *et al.* (2004) regarding the overturning wave and the effect of the shifting algorithm. The case is investigated at $t\sqrt{g/h_0} = 5.95$ which for the dimensions used in this study equates to 1.9s. The results are presented for the same configurations of the artificial viscosity and the density filtering algorithm used in the previous comparison. Simulations (a) to (d) again do not show the air phase for clarity.

Figure 5.23 shows two instances of the simulation without density filtering for different values of the artificial viscosity coefficient and use of the shifting algorithm. In the first instance this value is set to zero, causing the water particles to disperse. The dispersing is worse for the multi-phase cases, as the air particles cover the empty space. The air however, also allows the wave to be maintained, while in simulation (e) the toe collapses quickly.

The second column uses a value of α=0.03 for the artificial viscosity coefficient and as seen in Figure 5.23 it eliminates the dispersing of the particles. A side-effect of using the viscosity is a lower maximum height for the water flow. Compared with the BEM simulation, the height of the wave is much smaller especially when using the present multi-phase model. This leads to the creation of an air bubble sooner than it is expected by the BEM simulation.

Figure 5.24 shows the evolution of the flow for different values of the artificial viscosity coefficient, but using the density filtering algorithm. The density re-initialisation prevents the particles from dispersing even when no viscosity model is used. The results are also closer to the BEM results especially for the single-phase test case when no viscosity is used. As in Figure 5.23, using the artificial viscosity model results in a lower height for the overturning wave.

From Figure 5.23 and Figure 5.24 it is obvious that there are large differences between the BEM solution and the multi-phase simulations performed in this study. The multi-phase simulations by Colagrossi and Landrini (2003) show closer results to the BEM solution, while the single-phase results depend on the whether or not the artificial viscosity is used. In all cases, simulations (b)-(d) have a lower wave height and demonstrate a faster plunging wave trapping the air earlier than predicted by the BEM solution. These discrepancies occur for several reasons.

A primary difference is the boundary treatment. As seen in Figure 5.23 and Figure 5.24 in every DualSPHysics simulation there is empty space between the fluid particles and the boundary. This occurs upon the impact on the wall and it is not covered by air particles even when the shifting algorithm is used due to the boundary forces. For simulation (a), only the free surface position is provided and it is not possible to determine whether a similar occurrence exists. The boundaries were similarly responsible for creating a more prevalent particle spray as seen in the previous comparisons between the two simulations.

A secondary reason is the difference in the viscosity treatment since the simulations used in the present study do not use the viscosity reduction term by Balsara *et al.* (1995), the reason being the greatly increased computational cost especially in a three-dimensional space. As seen in Figure 5.23 and Figure 5.24 when including the viscosity model the maximum wave height is decreased. In addition, the plunging wave is breaking faster as can be clearly seen in Figure 5.24(e). This is not the case for the Colagrossi and Landrini (2003) simulations, where the height remains relatively constant with the viscosity having a smaller effect.

Of note is the difference in height between the single-phase and the multi-phase simulations. Even when the artificial viscosity is used the single-phase simulation maintains a higher flow height, although the plunging is breaking in the same manner. The difference is in the inclusion of the air particles and the exerted forces which prevent the unimpeded rise of the water flow. The air particles are also affected by the artificial viscosity, increasing their opposition to the water movement.

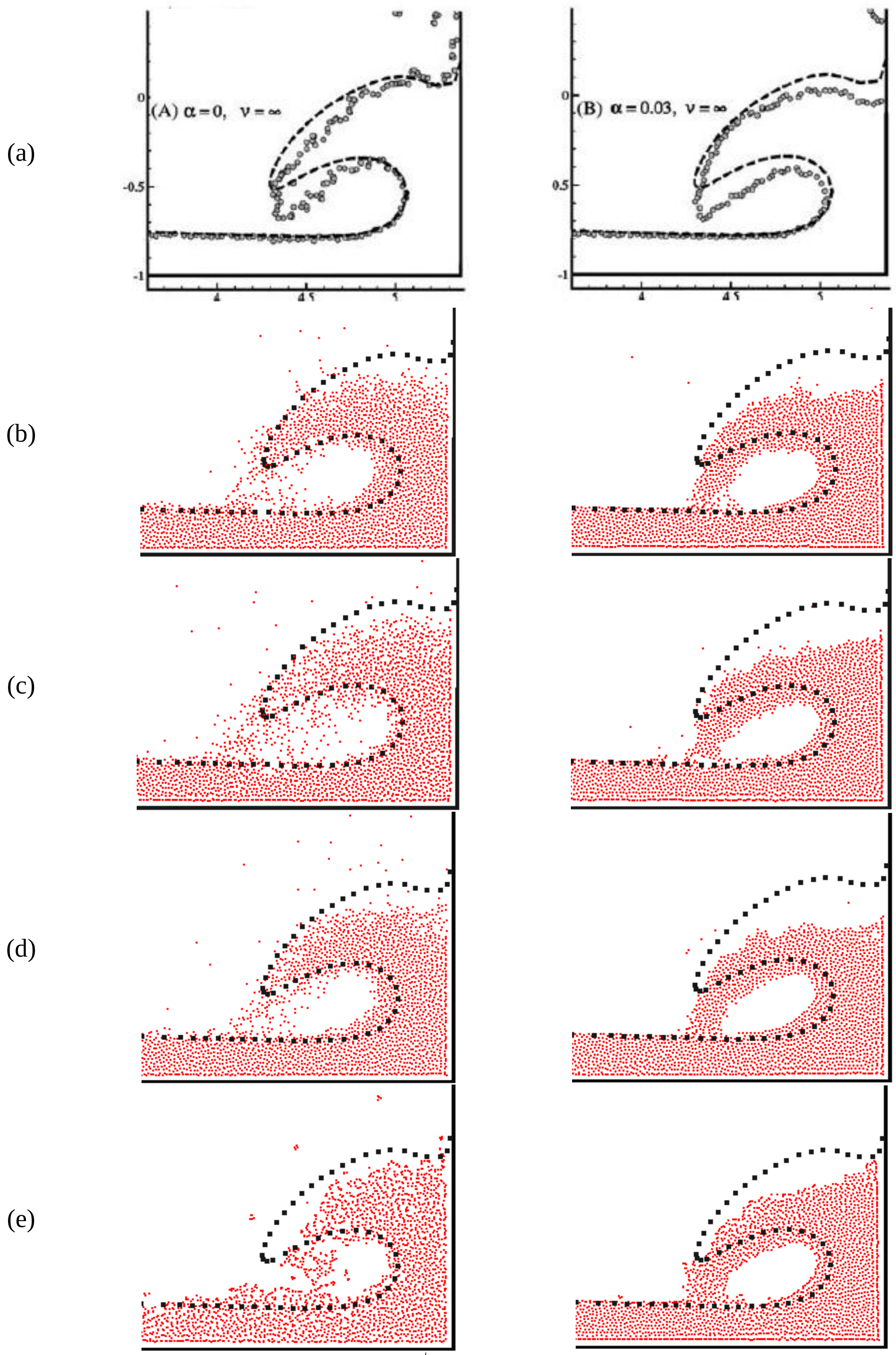


Figure 5.23:Comparison of the overturning wave at $t\sqrt{(g/h_0)}$=5.95 for dx/h_0=0.002 with a BEM solution (Faltinsen *et al.*, 2004) without density filtering for a) Colagrossi and Landrini (2003) b) present model without shifting c) present model with shifting only in the air phase d) shifting in both phases e) single-phase simulation. Simulations (a)-(d) do not display the air phase for clarity

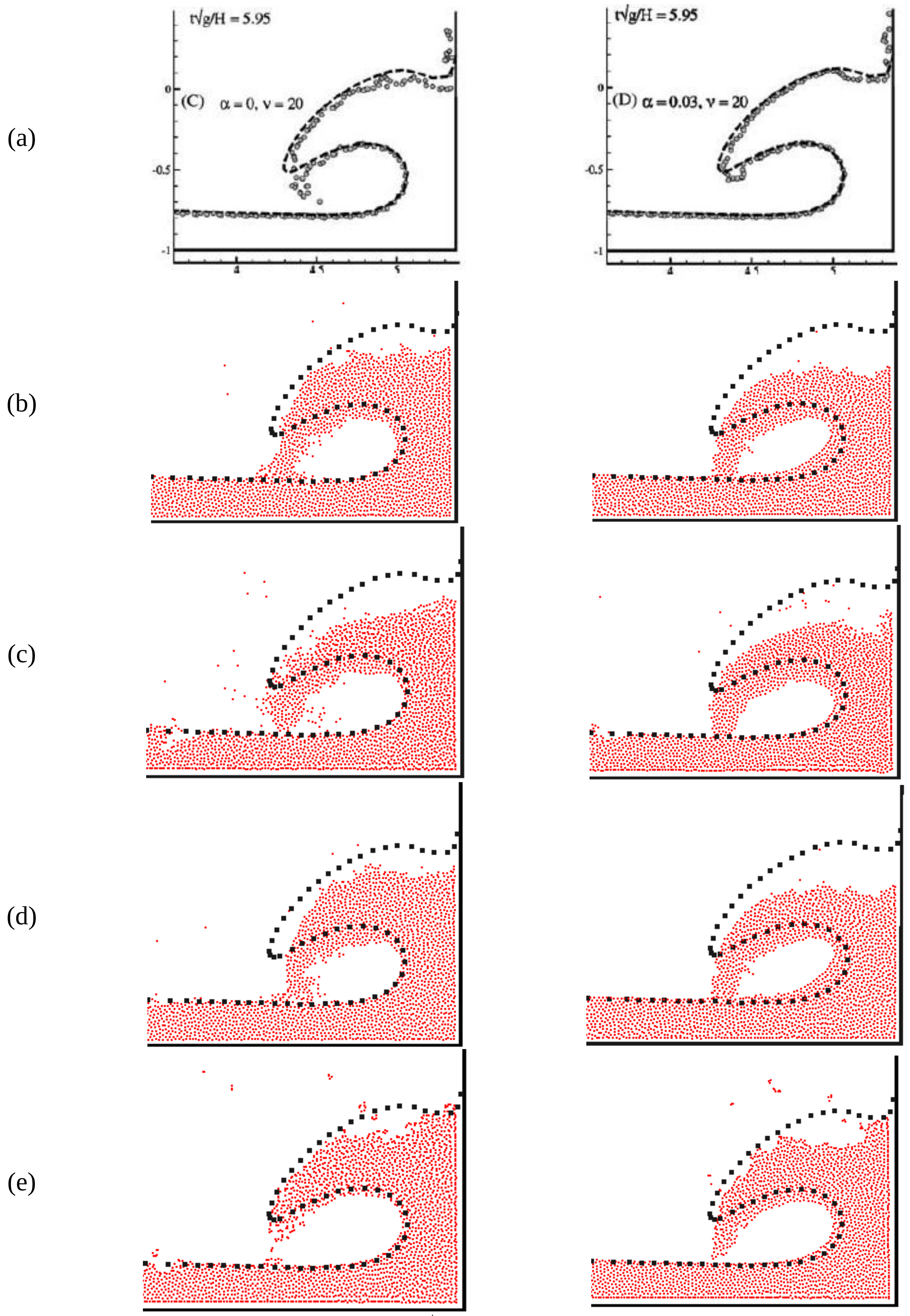


Figure 5.24:Comparison of the overturning wave at $t\sqrt{(g/h_0)}$=5.95 for dx/h_0=0.002 with a BEM solution (Faltinsen *et al.*, 2004) with density filtering for a) Colagrossi and Landrini (2003) b) present model without shifting c) present model with shifting only in the air phase d) shifting in both phases e) single-phase simulation. Simulations (a)-(d) do not display the air phase for clarity

A third reason is the difference in the density filtering, with one simulation using a first-order MLS filter and the other using a zeroth-order Shepard filter. The MLS filter is not implemented in the GPU code as it is expensive in terms of memory requirements. The MLS filter uses a more aggressive density re-initialisation scheme which adds to the numerical viscosity of the particles. The effect of this issue is lesser than the other reasons mentioned, but can be noticed when comparing the first instance of Figure 5.23 to the first instance of Figure 5.24.

It is clear that compared to the BEM results the viscosity of the particles in this study is larger. There exist several means with which to reduce it. The most direct one would the decrease of the artificial viscosity term, but as seen in Figure 5.23 even setting it to zero will not be enough for the multi-phase test case, although the single phase gives a good reproduction of the results.

Since the BEM solution is incompressible differences with the SPH simulation are expected. In order to diminish the effect of the air phase, the compressibility of the water particles can be varied by changing the value of coefficient *B* in the equation of state via the speed of sound. The ratio of the speeds of sound for the different phases can also be varied. Another important aspect that can be changed is the particle resolution, which, if increased, leads to a lower numerical viscosity for the particles.

A comparison with the BEM simulation (Faltinsen *et al.*, 2004) has been performed with a simulation with half the resolution increasing the particle number to 160,000. The new smoothing length is then $h/h_0 = 1.3\times10^{-2}$. Figure 5.25 shows the results for two values of the viscosity coefficient $t\sqrt{g/h_0} = 5.95$. The pressure field of the water particles is displayed, but the position of the air particles is also included to showcase some issues that appear when the shifting algorithm is not used. The issue appears in simulation (a) with some voids being created at the tip of the plunging wave. The voids only appear when using the artificial viscosity model.

Compared to the low resolution, the results for the new simulation are closer to the BEM simulation especially if the artificial viscosity model is not used. When the viscosity is used, the shape of the wave changes, becoming shorter and thinner than expected with its toe moving downwards at a higher speed. If it is not used, the shape of the plunging wave approaches the BEM solution with only the upper part of the wave being underestimated.

The shifting algorithm also affects the shape of the plunging wave. In simulation (c), when it is used for the water phase, the toe of the wave falls faster regardless of the artificial viscosity. The effect of the shifting algorithm can also be seen in simulation (b), even though it is not used in the air phase. Compared to simulation (a) while the position of the plunging wave is the same from a vertical perspective, its length is slightly shorter. The shifting algorithm then, increases the forces exerted in the plunging wave by the air particles and facilitates the downward movement of the water particles in the wave toe, thereby confirming the behaviour seen in Figure 5.23 and Figure 5.24.

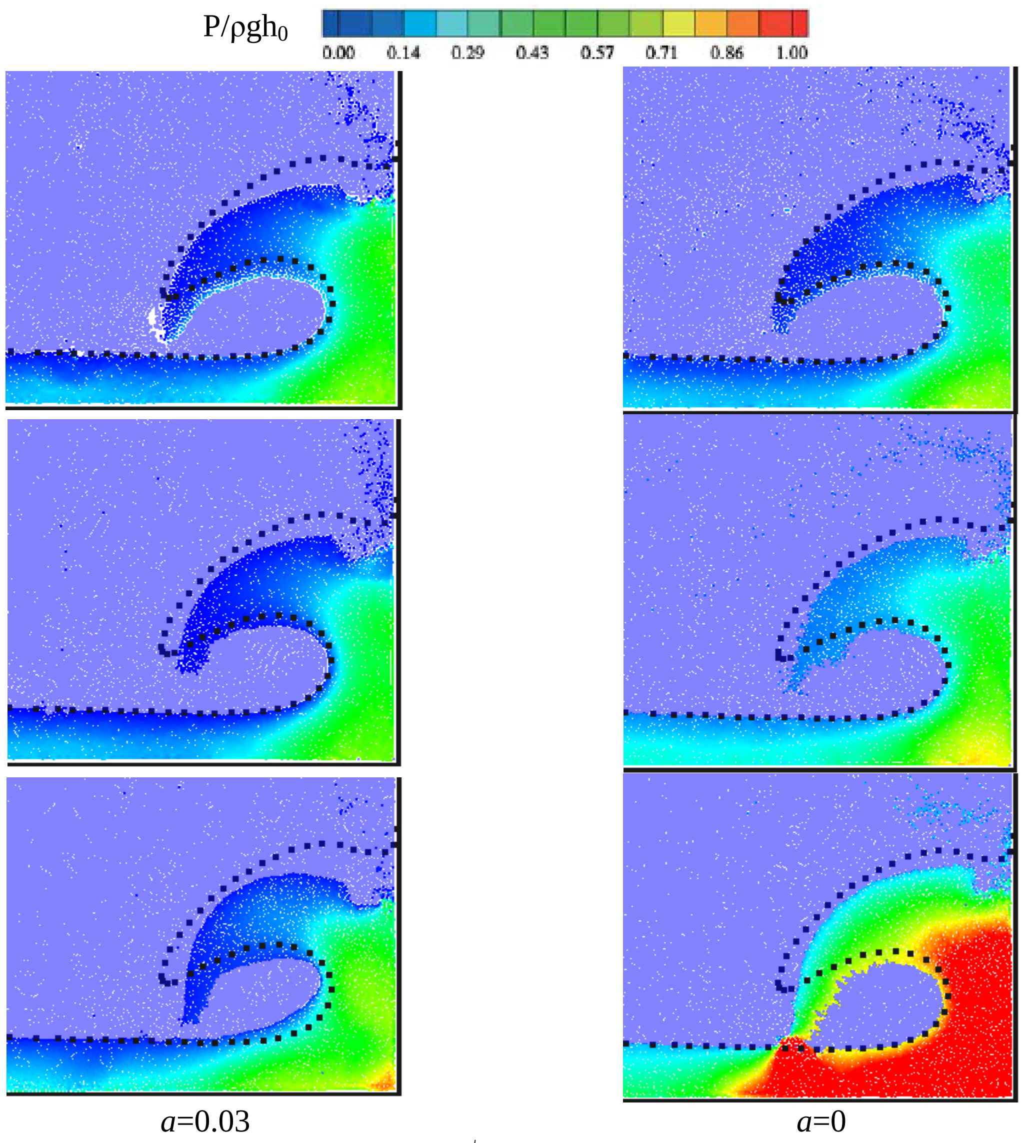


Figure 5.25:Comparison of the overturning wave at $t\sqrt{(g/h_0)}$=5.95 with a BEM solution (Faltinsen *et al.*, 2004) for a) present model without shifting b) present model with shifting only in the air phase c) shifting in both phases for dx/h_0=0.001

Regarding the pressure field, the increased pressure observed in Figure 5.22 is also present in Figure 5.25. Simulation (c) shows an increased pressure field, although only slightly if the artificial viscosity model is used. Eliminating the viscosity on the other hand, leads to increased pressure for simulation (b) as well although at a lesser degree. For the latter case the increase is more apparent in the area near the free surface.

In general, there are several behaviours which we can deduce from the previous comparison. The viscosity of the particles is a major factor in predicting the shape of the flow especially when discussing the reflected wave. Low resolutions have increased numerical viscosity that greatly affects the overturning wave results. Special reduction terms such as the one by Belytschko *et al.* (1998) can be used to limit its effects. Increasing the resolution reduces that source of error leading to closer results compared to an incompressible BEM solution. Apart from the compressibility, differences with the BEM method can also be attributed to the different boundaries each method is using.

The artificial viscosity term, at least with the coefficient value used, increases the viscosity of both phases significantly, leading to errors in the plunging wave without greatly improving the pressure field. The latter is achieved by the density renormalisation term. In lower resolutions, using the artificial viscosity allows for a more clearly defined free surface; its effect however, is diminished for the higher resolution.

The shifting algorithm is necessary for maintaining a void-free field for the higher resolutions. Its inclusion however, does alter the shape of the overturning wave. It also affects the pressure field of the simulation leading to its increase. The effect initially occurs in the air phase, as evidenced by the affected free surface pressure in Figure 5.22 and Figure 5.25 and then spreads to the water phase. The increase is diminished if the artificial viscosity model is used, with both Figure 5.22 and Figure 5.25 showing a similar free surface pressure across all simulations.

Apart from the comparison with the BEM test case the results can be compared with a Level-Set algorithm by Colicchio *et al.* (2005), which follows the evolution of the overturning wave. The comparison will be done for two instances, at $t\sqrt{g/h_0} = 6.76$ and at $t\sqrt{g/h_0} = 7.14$ displayed in Figure 5.26 for the first instance and Figure 5.27 for the second one. The simulations used are the same presented in Figure 5.25.

Compared to the Level-Set algorithm there is generally good agreement with the SPH simulations. The agreement is better for simulation (b), which uses the shifting algorithm only for the air phase. When using the artificial viscosity, the air bubble is very well predicted but the splash-up is lower than expected by the Level-Set algorithm. The situation is reversed without the artificial viscosity; the splash-up occurs at the same point but there are some differences between the Level-Set algorithm and the SPH simulation in the bubble.

Without the shifting algorithm in simulation (a), the results are rather similar to simulation (b), although the agreement with the Level-Set algorithm is not as good with the height of the splash-up being underpredicted and the bubble having a smaller volume. The greatest issue appears when using the artificial viscosity. As seen in Figure 5.25 voids are being created in this case as a result of the increased velocity of the splash-up. If the shifting algorithm is used for the water phase as in simulation (c), the agreement with the Level-Set algorithm is better for the splash-up but worse for the bubble. The volume and especially the position of the bubble are not as close as in the other two simulations. Using a viscosity model leads to changes in the upper part of the flow; the water volume over the air bubble decreases significantly. The splash-up is longer and less dispersed than the other cases, likely due to the effect of the free-surface term.

Regarding the pressure field, the increased pressure seen in Figure 5.25 can also be observed in Figure 5.26. The increase is greater for the cases with the shifting algorithm but without the artificial viscosity, although a minor increase can still be seen for simulation (c). The increase is greater near the free surface, indicating that the air pressure has increased.

In general, increasing the viscosity of the flow results in a lesser height of the water splash-up and a slightly shorter plunging wave. The shifting algorithm in the water in conjunction with the viscosity model leads to issues when predicting a suspended flow. On the other hand, shifting in the air phase only seems to have an effect in the area around the bubble, leaving the main water flow unaffected. A common issue among the simulations is under predicting the height near the right wall.

The conclusions derived from the previous graphs hold true if the evolution of the flow is examined, as seen in Figure 5.27 which presents the flow at the later time of $t\sqrt{g/h_0} = 7.14$ for the same values of artificial viscosity. Figure 5.27 shows a good agreement between the Level-Set algorithm by Colicchio *et al.* (2005) and the SPH simulations, especially for simulation (b). The results for simulation (c) are similar but the splash-up is better defined although the volume in the upper part of the flow is underestimated.

From the second column of Figure 5.27 without the viscosity model, the same conclusions drawn from Figure 5.26 can be reached. The voids present without the shifting algorithm persist and the splash-up has a smaller height than expected but the bubble is captured very well. A lower pressure field compared to Figure 5.27 is also present with a slight rise only observed for simulation (c). The latter shows significant issues when dealing with the water flow over the air bubble. There is only a very thin layer in that position and the air bubble will burst soon, which is an unexpected behaviour.

In general, the shifting algorithm in this case has a significant effect in the flow properties. If used only for the air phase it prevents the formation of voids and improves the simulation around the bubble area, without obstructing or altering the main water flow. However, if used for the water phase, the particle positions change leading to errors in the simulation. The differences are better observed in Figure 5.27 without the effect of the viscosity model and are mostly situated around the bubble and in the splash-up region.

It should be noted that this behaviour might be improved with a different SPH multi-phase algorithm but as noted previously, they are more difficult to implement on a GPU and will be the topic of further work. The shifting algorithm also affects the pressure of the fluids resulting in an increase. This behaviour occurs for both the water and the air phases and appears very early in the simulation as seen in Figure 5.22. The behaviour occurs because the shifting algorithm is affecting the compressibility of the fluids altering the density values in the process. Different values and ratio are required for the speeds of sound of the different fluids to reproduce an identical pressure field.

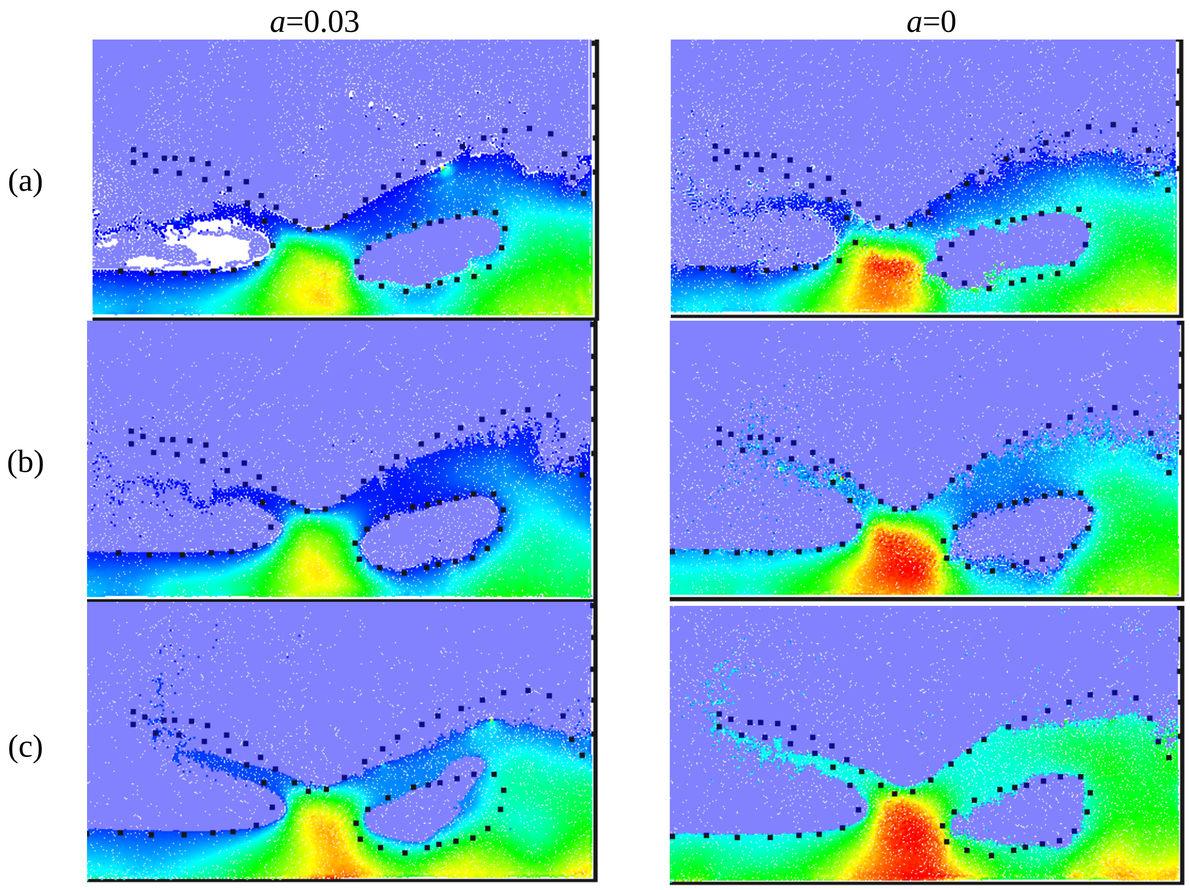


Figure 5.26:Comparison with a level-set algorithm (Colicchio *et al.*, 2005) for a) present model without shifting b) present model with shifting only in the air phase c) shifting in both phases at $t\sqrt{(g/h_0)}$=6.76 for dx/h_0=0.001

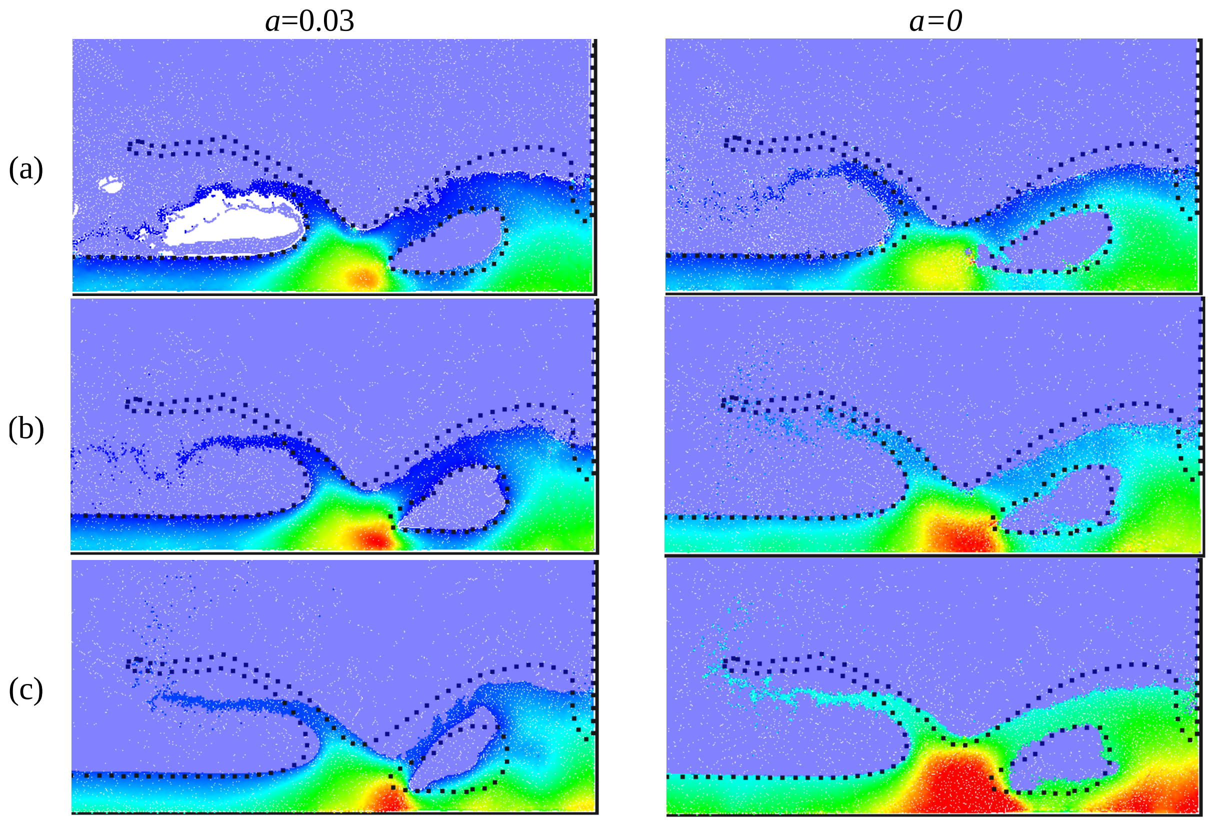


Figure 5.27:Comparison with a level-set algorithm (Colicchio *et al.*, 2005) for a) present model without shifting b) present model with shifting only in the air phase c) shifting in both phases at $t\sqrt{(g/h_0)}$=7.14 for dx/h_0=0.001

5.4. Wet Dam Break

5.4.1. Case description

This case is similar to the previous dam break case, but the major difference is the existence of a shallow layer of water present continuously in the vessel. It was selected as an example of a more complex flow with mixing both between the two phases and between the two bodies of water. The case is based on the experiments of Janosi *et al.* (2004) and is also characterised by the presence of a lock gate separating the main body of the water from the thin layer. The lock gate was placed in a fixed position, but experiments with different water heights were performed. A schematic arrangement is shown in Figure 5.28 where d_0 is the height of the water layer and h_0 is the height of the main water body.

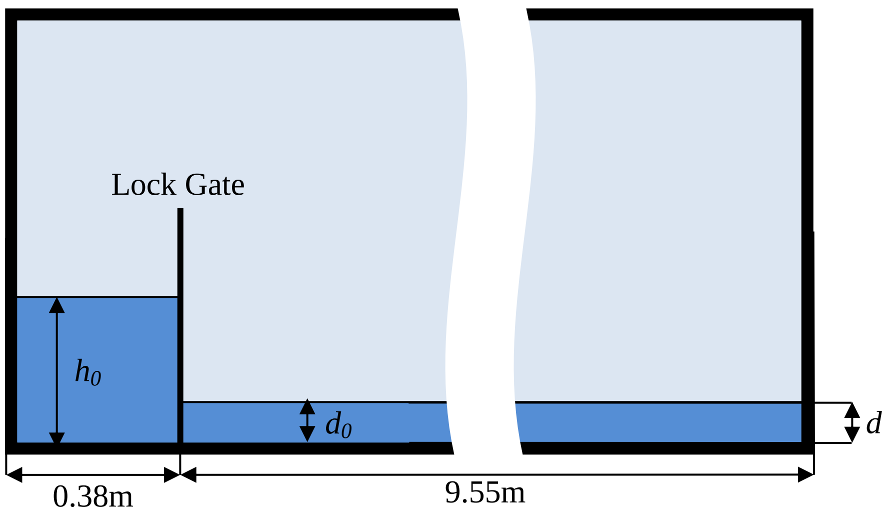


Figure 5.28: Definition Sketch for Wet Bed Dam Break Case

The behaviour of the water flow is drastically different in the presence of the thin layer. Releasing the lock gate leads to the main body forming a propagating bore, similar to the previous case. However, the thin layer remains quiescent and resists the movement induced by the height difference between the two bodies of water. The movement is then caused by the pressure as well as the effect of the gravity. After the initial acceleration, when the pressure is equalised, the flow is initially maintained due to inertia, but the kinetic energy quickly dissipates and the water motion stops.

In particular, the wet bed leads to the creation of a ‘mushroom’ formation as the propagating bore is being formed at the start of the simulation. This formation has also been reported by

Stansby *et al.* (1998). This leads to unstable configurations and surface breaking in both forward and reverse directions (Stansby *et al.*, 1998).

Due to the resistance the effect of the gate is not immediately eliminated as in the dry bed case. The speed of the gate is the same as the wave celerity; as a result, the removal of the gate needs to be simulated. It only affects the simulation for the first 0.1s, but it actively prevents the free flow of the water in the lock.

Janosi *et al.* (2004) proposed that the effect of the side walls on the flow inside the tank, at least in the earlier stages of the propagation, is negligible and therefore, the case can be modelled in a two-dimensional domain. The air phase also has an important effect on the behaviour of the propagating wave, as the surface breaking results in the entrapment of air bubbles, changing the speed, pressure and form of the wave. Depending on the depth of the water layer, the surface breaking can occur multiple times until a smooth bore similar to the previous case can be formed.

The present simulation will be a recreation of the experiment. The lock gate will be modelled using fictitious boundary particles and an initial constant velocity. To reduce the size of the domain, the tank will be completely closed. The height of the tank is three times larger than the water height, to ensure that the limited domain does not interfere with the wave propagation. The experiment was initially ran in order to compare it to a flow of mixed water and polymer additive but only the results for the clean water flow will be used in this study.

Important factors in this case are the initial height of the water volume left of the lock gate, h_0 and the ambient fluid depth in the channel, d_0. Profiles were presented for h_0=15cm but for different fluid depths 0 (dry bed), 18mm and 38mm. The dry bed flow was reported to have a Reynolds number about 30,000 to 40,000 meaning that a turbulent boundary layer was present at the bottom of the vessel. The water height behind the front was used as a characteristic length for the calculation of the Reynolds number (Janosi *et al.*, 2004).

The case has been simulated with SPH by other researchers (Crespo *et al.*, 2008) (Gomez-Gesteira *et al.*, 2010). It has not so far been simulated using a multi-phase model. Khayyer and Gotoh (2010b) compared incompressible and compressible SPH and MPS models with the experiment profile. They found that the backwards breaking of the free-surface and the resulting circulation could not be reproduced by weakly compressible SPH models without modification of the viscosity term. In their study, to reduce the flow decay in areas of high vorticity a modified version of the reduction function by Balsara (1995) was used, presented in Equation (5.24). It is not used in the current study, due to its high computational demands.

$$f_{ij}^{vr} = \frac{2\left|\nabla \times \mathbf{u}\right|_{\max}}{2\left|\nabla \times \mathbf{u}\right|_{\max} + \left|\nabla \times \mathbf{u}\right|_{i} + \left|\nabla \times \mathbf{u}\right|_{j}}. \tag{5.24}$$

5.4.2. Differences between the experimental and numerical results

Figure 5.29 shows an instance of the simulation for the low water layer at 0.28s. It can be observed that the wave being created is at a different point in the tank than the experimental result. In addition, the water flow is not at the same instant with the experiment at this time. The overturning wave has already been formed in the simulation, while in the experiment the crest is only being formed.

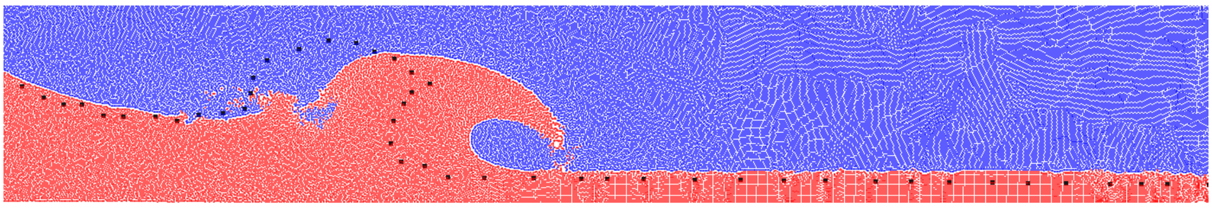


Figure 5.29: Discrepancy between experimental and numerical results

This behaviour is present for both the multiphase and a single-phase simulation. The discrepancy between experimental and numerical results has been observed for most SPH simulations regarding this case (Lee *et al.*, 2008) and has been measured to be about 43ms (Violeau and Issa, 2007b). The reason for this difference is related to the gate movement. The velocity of the gate as it is rising is stated to be 1.5m/s. However, due to inertia effects this velocity cannot be reached instantaneously, meaning that the time needed for the removal of the gate is slightly longer (Violeau and Issa, 2007b) creating the discrepancy observed here.

The time given by Violeau and Issa (2007b) for the delay caused by the gate agrees with the simulations performed for this study. Figure 5.30 shows the same case from Figure 5.29, but the screenshot has been taken at 0.24s; the results are much closer to the experiment with the water flow being at the correct position. However, simply changing the comparison time will still introduce a small error as the gate acceleration is not considered. Approximating this acceleration is impossible due to the lack of available data; the lock gate is assumed to gain its terminal velocity immediately.

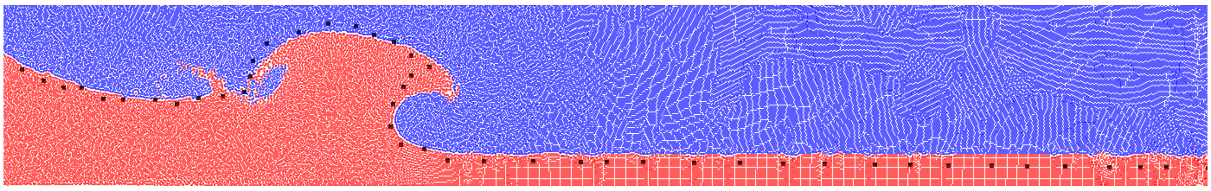


Figure 5.30:Screenshot of the wet dam break instance 43ms earlier

From Figure 5.29, it is noticeable that the wave of the experiment and the simulation are in different instants. Khayyer and Gotoh (2010b) mention that the wave appears earlier in the simulation and that the flow must be shifted in space (shifting value depends on the time but 0.02m is an average value) in order to overlap the experimental results. An example can be seen in Figure 5.31 showing the corresponding wave phase. This is an issue primarily originating from the incorrect simulation of the gate movement. Apart from the discrepancy of the gate movement, Khayyer and Gotoh (2010b) and Issa and Violeau (2009) have also cited the SPH viscosity models and their dissipation of the vorticity field as being a factor in the inaccuracy of the numerical results.

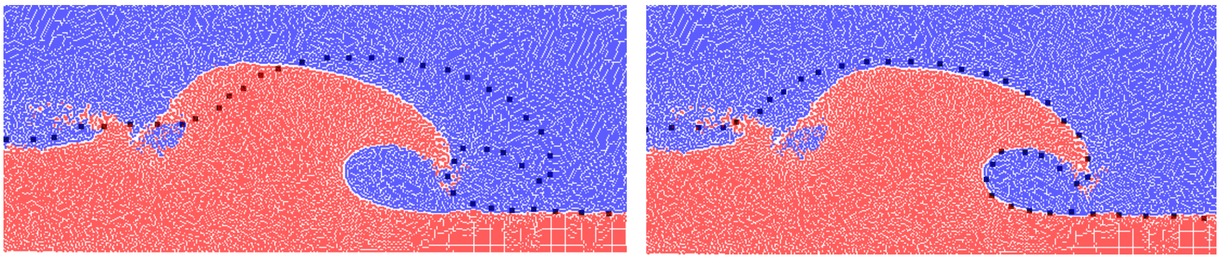

(a) (b)

Figure 5.31: (a) Phase difference between the multi-phase SPH simulation and the experimental results (b) Results after shifting the particle position according to Khayyer and Gotoh (2010b) for *dx/h_0*=0.0067

However, Crespo *et al.* (2008) did not mention an issue with the phase between the experimental and the numerical results. In their simulation they used a value for the artificial viscosity coefficient equal to 0.08 and non-dimensionalised the particle spacing *dx* according to the initial water column height h_0. The resolution was relatively large, set as dx/h_0=0.0333. Their simulation was executed with the single-phase code and showed little difference between the numerical and experimental results, as seen in Figure 5.32. The need for delaying the simulation by 0.043s however, remained. However, Figure 5.32 shows that a multi-phase using the same resolution as the single-phase simulation produces closer agreement with experimental data.

Single-phase DualSPHysics

New Multi-phase DualSPHysics

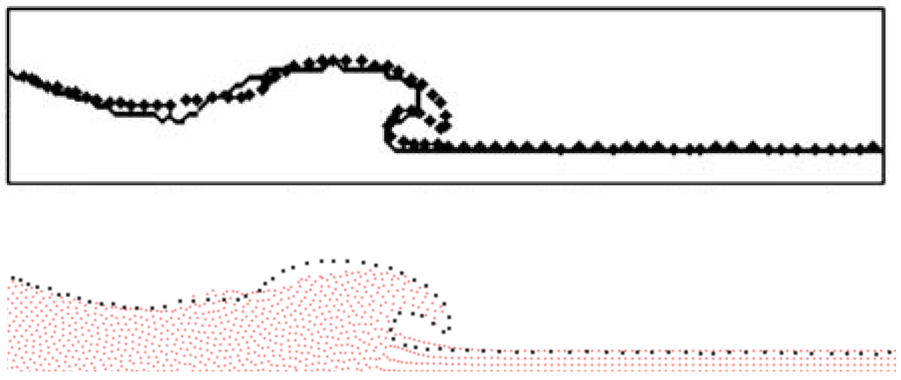


Figure 5.32: Comparison of the single-phase SPH results of Crespo *et al.* (2008) with the multi-phase SPH. Both simulations use the same resolution *dx/d_0*=0.0333

The reason for the difference between the two simulations can be traced to the number of particles being used. The numerical viscosity for the lower number of particles is higher causing a stalled movement of the flow. In contrast, using a greater number of particles reduces numerical viscosity that may be present causing quicker breaking of the plunging wave. Hence, it can be concluded that choosing the correct time shift to match numerical and experimental results and include the effect of the moving gate is not straightforward.

5.4.3. Low water layer: d_0=0.018m

Figure 5.33 to Figure 5.37 (pp.179-181) present the results for the lower water layer at the tank, where the water layer height is 0.018m and the ratio of the water tank height over the thin water layer 0.012. This case is one of the SPHERIC benchmarks (Crespo *et al.*, 2008) and screenshots from the experiment of Janosi *et al.* (2004) have been digitised by Crespo *et al.* (2008). The results of the simulation will be compared to the digitised points as well as the previous SPH simulations.

A slight issue arising from the digitised points is the inconsistency on the depiction of the water level at the right side of the tank. The water there is undisturbed and its height should remain constant before the arrival of the wave, however, the digitised points show a difference in height that can be up to 4mm. For the comparison, the still water level of the simulation will then be set to be at the average of the different levels; however this may introduce some errors when comparing the wave height. For reference, the original screenshots of the experiment by Janosi *et al.* (2004) will also be provided.

The computation has been run with different resolutions. The first instance uses particle spacing *dx/h_0*=0.0133 which is the same as Khayyer and Gotoh (2010b). The artificial viscosity coefficient used is also the same as that study, however, no information regarding the speed of sound used is given. Crespo *et al.* (2008) propose the speed of sound to be based on Equation (5.4), giving a value of 15m/s. Violeau and Issa (2007b) measured the maximum speed in their simulation and found that the appropriate value should be 25m/s. In this study, it was found that the latter value gives results closer to the experimental data for the resolutions used and will be used in the following figures. As discussed in the previous section, the movement of the gate is delayed for 0.043s while the artificial viscosity coefficient is set at 0.01s.

Figure 5.33 and Figure 5.34 show the evolution of the water flow for the first half of the wet dam break experiment. The low speed with which the gate is raised causes the creation of a wave breaking backwards, trapping a small amount of air inside the water flow. The forward motion is then creating a plunging breaker.

As expected, for the first instances there are only small differences between the single and the multi-phase simulation. Both simulations are able to predict the back-breaking wave without the use of Equation (5.24), the vorticity reduction function by Khayyer and Gotoh (Khayyer and Gotoh, 2010b). It is more accurately predicted by the multi-phase simulation, which shows the mixing between the two phases. For the plunging breaker the multi-phase model correctly predicts the shape of the wave but the toe position is slightly lower than expected.

Figure 5.35 to Figure 5.37 show the latter half of the simulation where the differences between the different simulations become more pronounced. The plunging breaker traps a significant volume of air and subsequently creates a splash-up evolving in another plunging wave. The simulation and the available experimental data finish before the breaking of this wave. Both simulations have satisfactory agreement with the experimental results.

The biggest difference is in the splash-up and the breaking wave that is being created. As we can see in Figure 5.35 the height of the splash-up wave is higher than expected, a trend that continues in Figure 5.36 and Figure 5.37. There is however, clear improvement for the multi-phase model when predicting the splash-up wave although the water spray and the phase mixing is not as prevalent as the experimental results.

Regarding the entrained air in the water flow, Figure 5.35 shows that both simulations can predict its behaviour correctly right after its creation. However, the single-phase simulation cannot predict the shape and volume of the air bubble as the simulation progresses and eventually the void inside the water flow is entirely eliminated as shown in Figure 5.37. This is not the case for the multi-phase simulation, where the addition of the air particles allows the air phase to retain its volume inside the water flow and the simulation to predict the position and volume of the bubble.

The advantage of modelling the air particles can also be seen at the back of the plunging breaker, where the two phases are mixed. While the single-phase test case is able to initially predict the air bubbles that appear as seen in Figure 5.34, they quickly vanish with the water particles covering the voids. In the multi-phase code on the other hand, the two phases remain mixed until the end of the computation as seen in Figure 5.37. Moreover in Figure 5.37 the advantage of using the multi-phase simulation is demonstrated with closer agreement for the splash-up predicted.

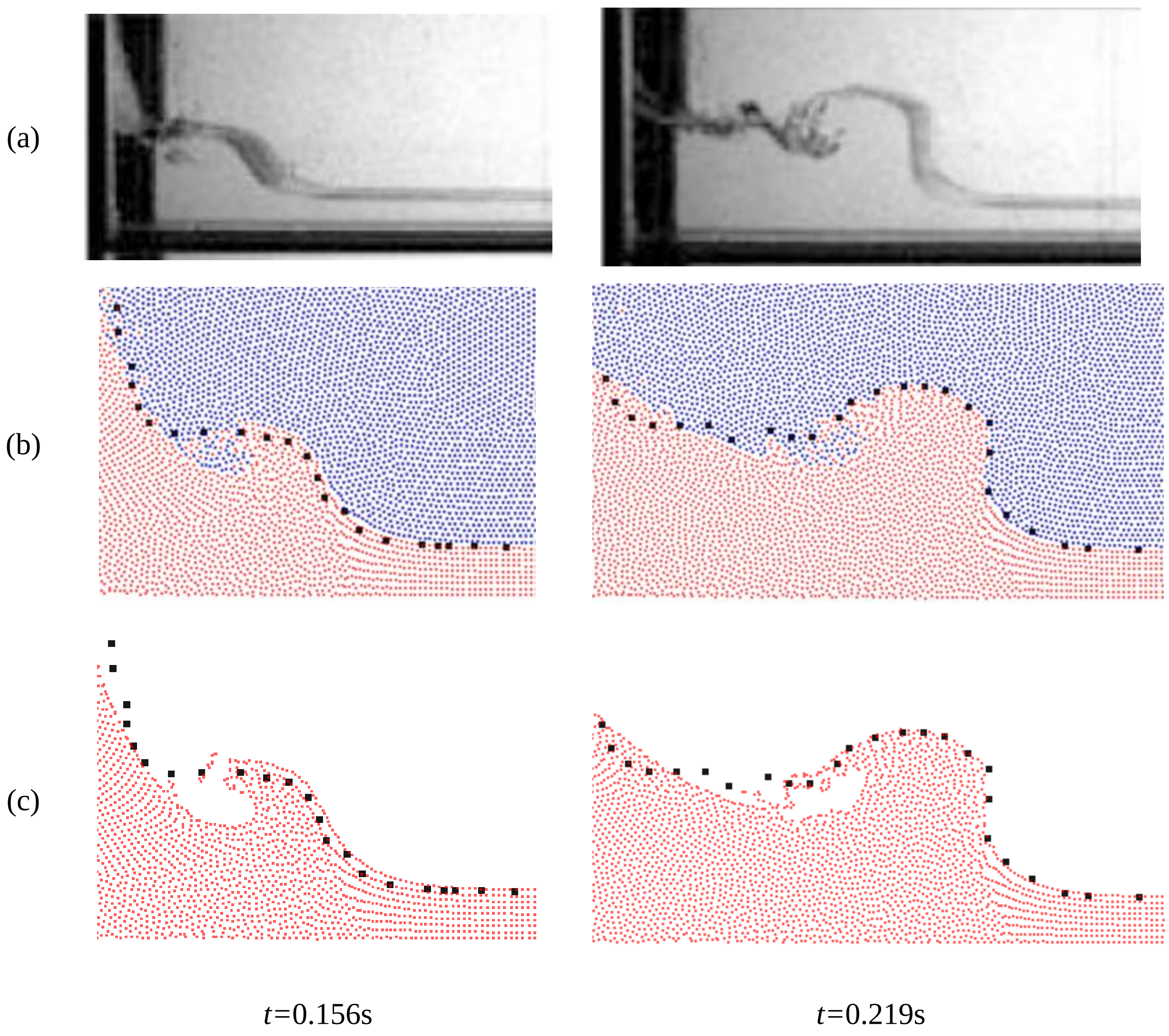


Figure 5.33: Comparison of the experimental results to a single and a multi-phase simulation for t=0.156s and t=0.219s for resolution dx/h_0=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

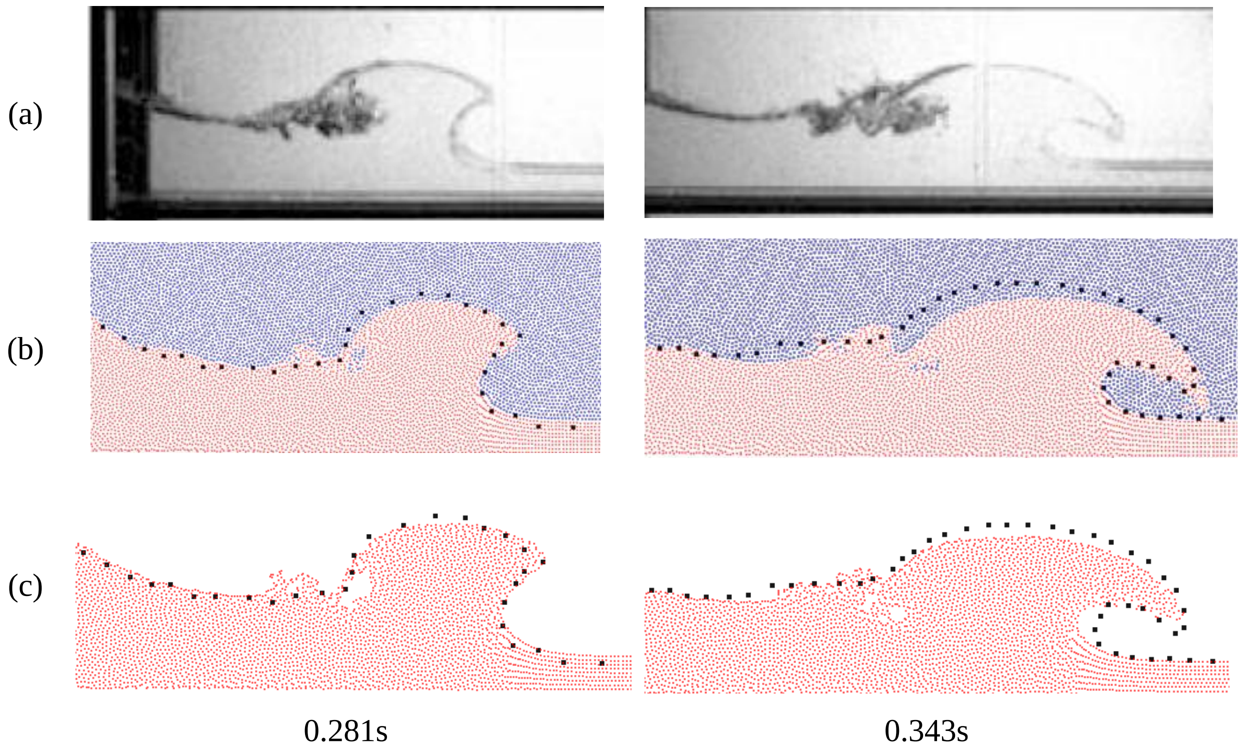


Figure 5.34: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.281s and *t*=0.343s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

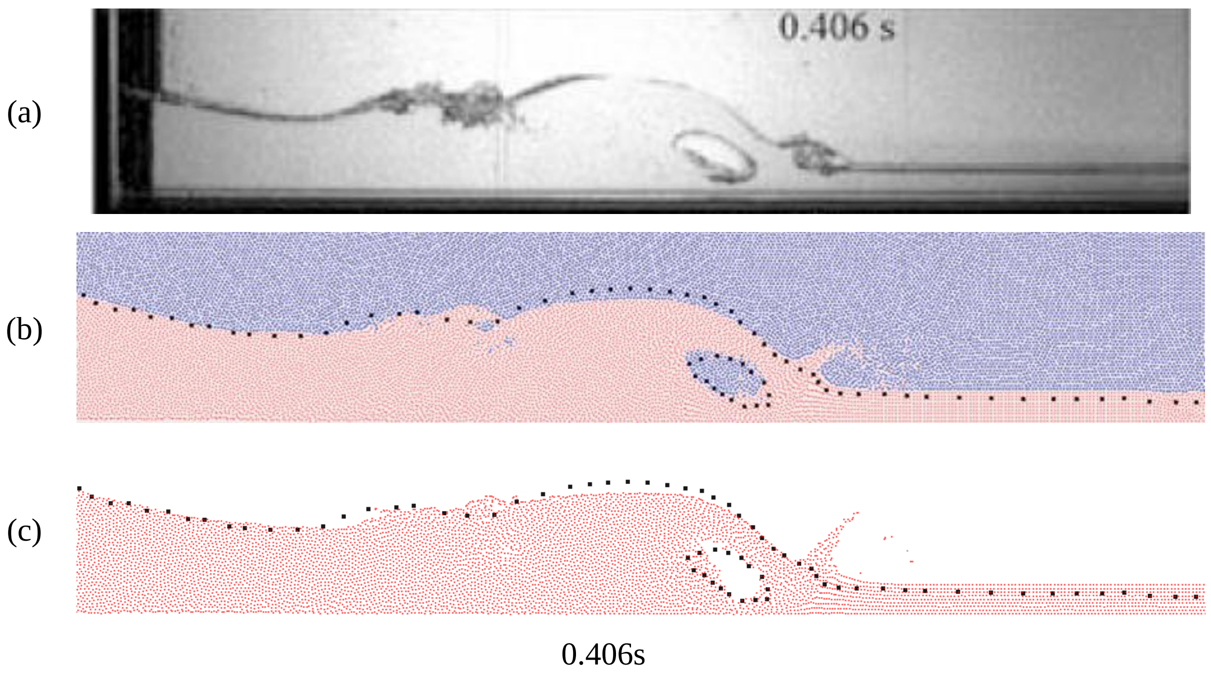


Figure 5.35: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.406s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

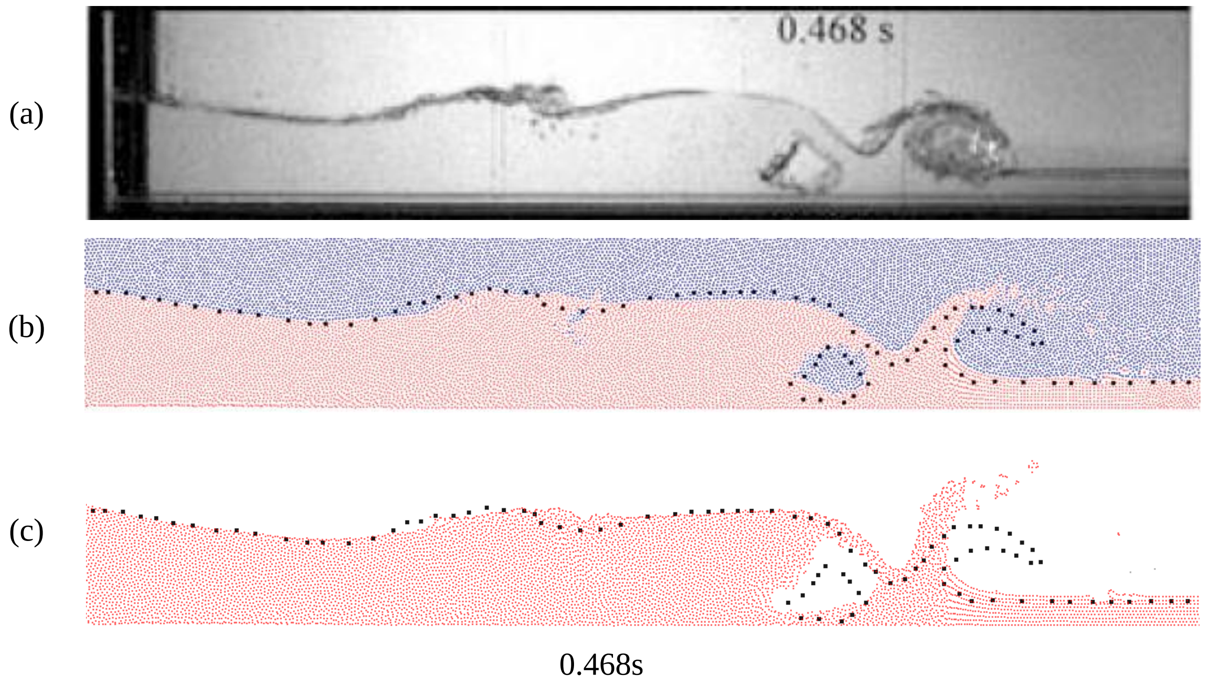


Figure 5.36: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.468s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

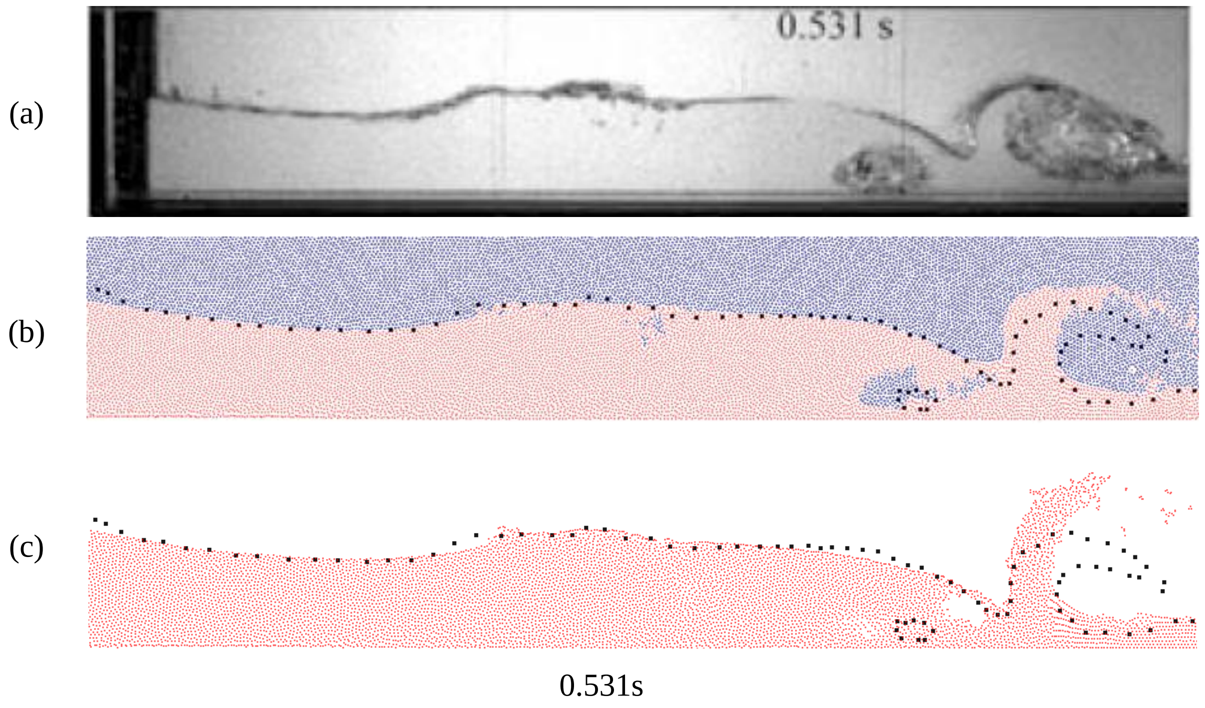


Figure 5.37: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.531s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

The case has also been executed for a finer resolution, with particle spacing dx/h_0=0.0067, doubling the resolution of the previous simulation. The speed of sound and the artificial viscosity are the same as the previous simulation. The results are presented in the same manner, comparing the experimental results to a multi-phase and a single-phase simulation. The instance from the simulation occurs 0.043s earlier following the phase difference between the experimental and numerical results discussed earlier as outlined by Violeau and Issa (2007b). The results are displayed in Figure 5.38 to Figure 5.42 (pp. 183-185).

The higher resolution shows more differences between the single and the multi-phase simulation. Figure 5.43 shows that the single-phase model is closer to the digitised points but comparing the experiment screenshot to the simulations, we can see that the back-breaking is captured better in the multi-phase simulation, due to the physical presence of the air particles, which allows for mixing between the phases.

Figure 5.44 shows the evolution of the water flow at 0.281s and 0.343s. The behaviour seen in Figure 5.38 is repeated here: the multi-phase simulation is slightly underestimating the height of the flow but the air-water mixing is better represented. Both simulations progress faster than the experiment. The issue has been also observed by Violeau and Issa (2007b) and Khayyer and Gotoh (2010b) who identify the numerical modelling of the viscosity and the vorticity to be the cause of the issue.

Between the two simulations, the multi-phase model also seems to be progressing faster than the single-phase simulation as seen in the tip of the plunging wave. The reason for the difference is the additional force exerted on the wave by the air particles, which increases its falling velocity.

The faster progress of the simulations can also be seen in Figure 5.40, where the splash-up created by the breaking wave is more advanced. Apart from that issue, the experiment is well captured by the simulations including the entrained air bubble created by the plunging wave. The multi-phase model is also capturing the air-water mixing observed at the back of the flow as the air particles trapped during the back-breaking maintain their position inside the flow. In contrast, the voids present in Figure 5.39 in the single-phase simulation have all but vanished.

The results presented in Figure 5.41 and Figure 5.42 follow closely the results presented for the lower resolution. The modelling of air particles allows for the prediction of the evolution of the air bubble; the void in the single-phase is eventually diminished as seen in Figure 5.42. The presence of air also assists in modelling the splash-up; while the spray is not reproduced the forces exerted from the air particles produce a splash-up closer to the experimental data.

In contrast the single-phase splash-up continues its motion unobstructed producing a plunging wave of much greater height.

Compared to the low resolution case, these simulations give very similar results. Differences are mostly seen in the backward breaking area, which is better resolved with a larger number of particles. Of particular note is the behaviour of the simulations at the plunging wave; the results for the lower resolution for its toe are closer to the experimental ones. This occurs due to the numerical viscosity associated with the lower number of particles as seen by the simulations of Crespo *et al.* (2008).

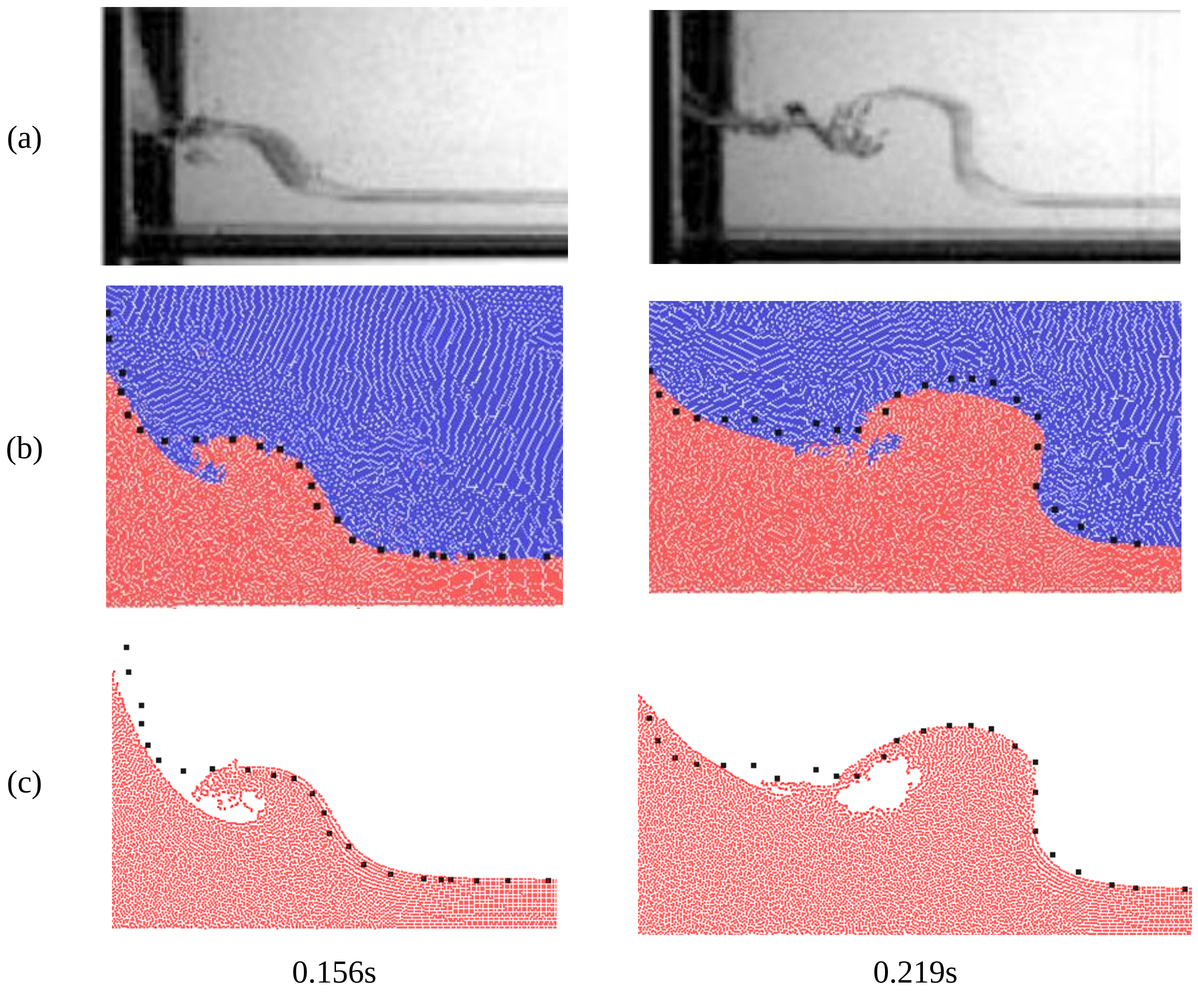


Figure 5.38: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.156s and *t*=0.281s for resolution *dx/h*$_0$=0.0067. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

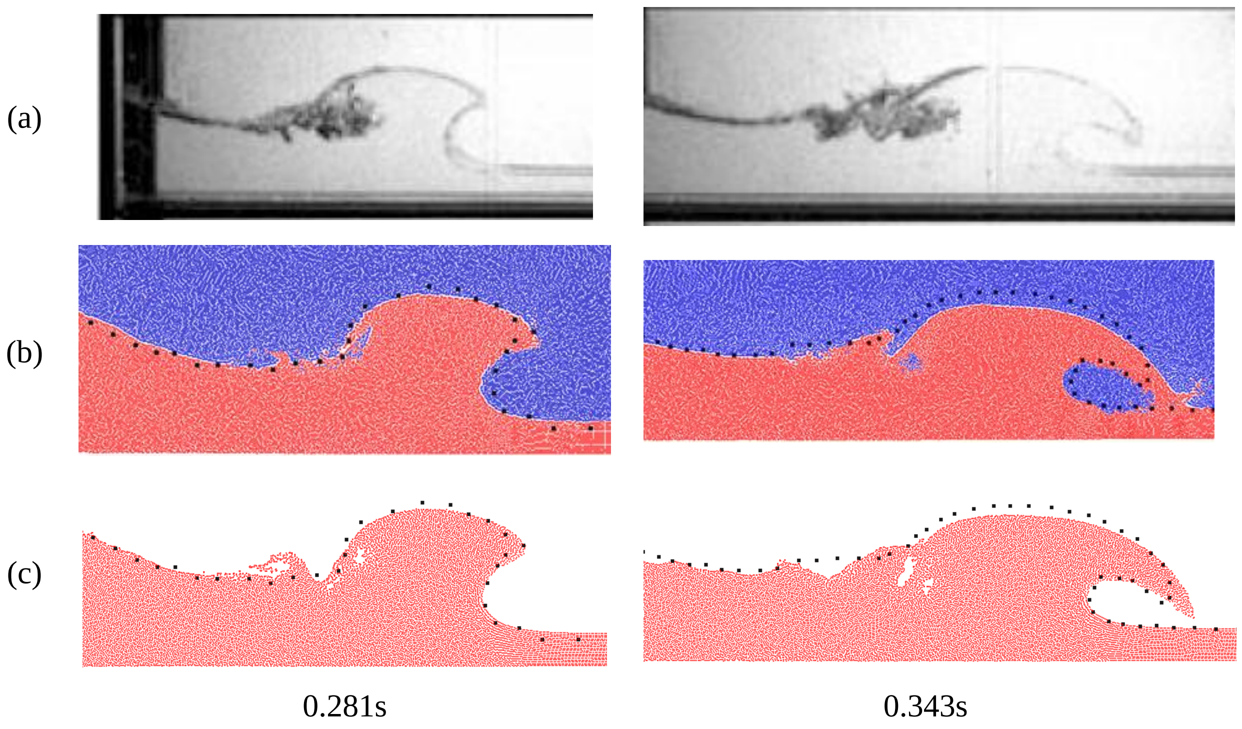


Figure 5.39: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.281s and *t*=0.343s for resolution dx/h_0=0.0067. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

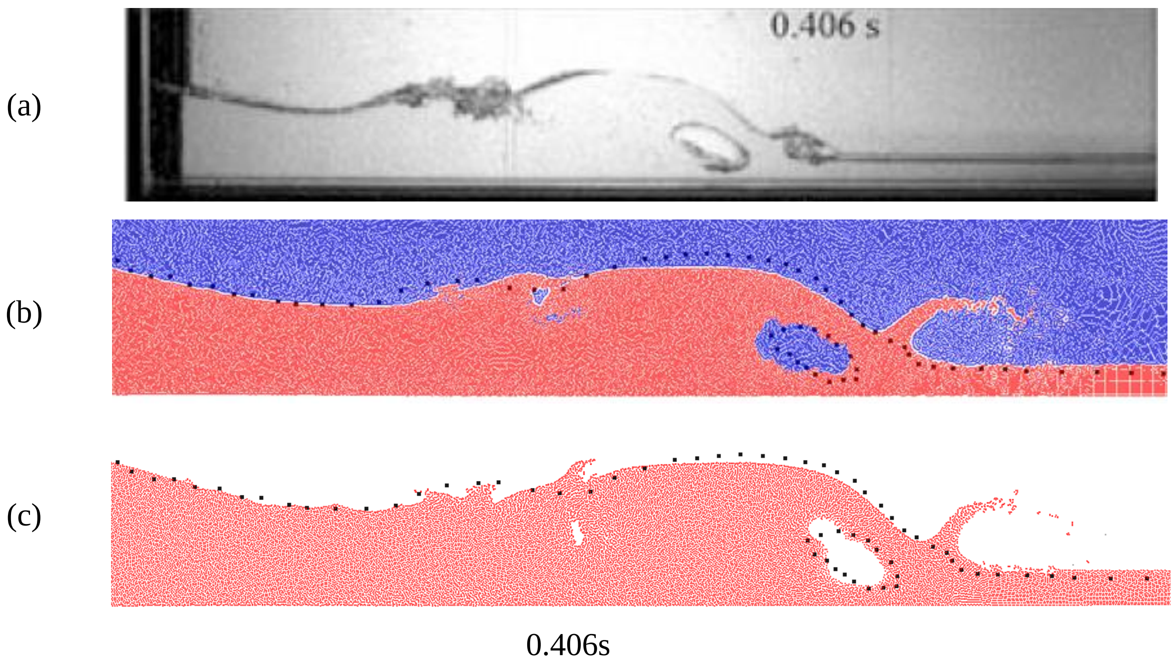
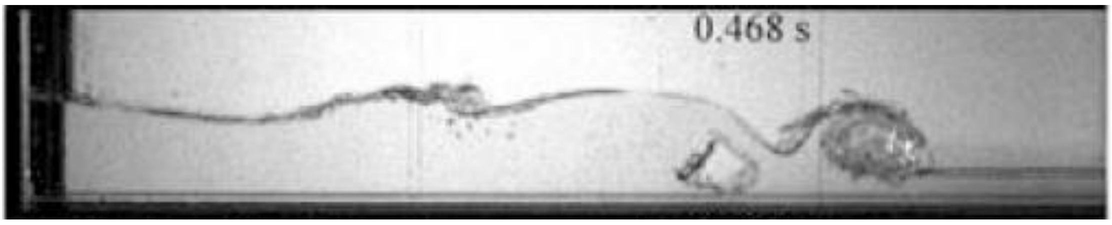
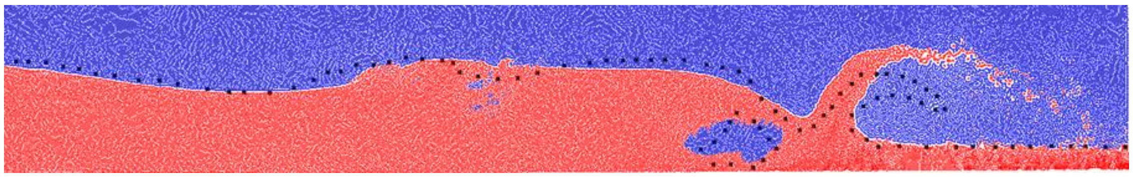


Figure 5.40: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.406s for resolution dx/h_0=0.0067. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

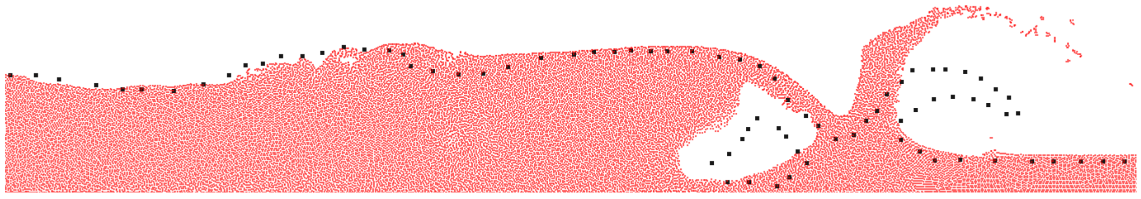
(a)



(b)



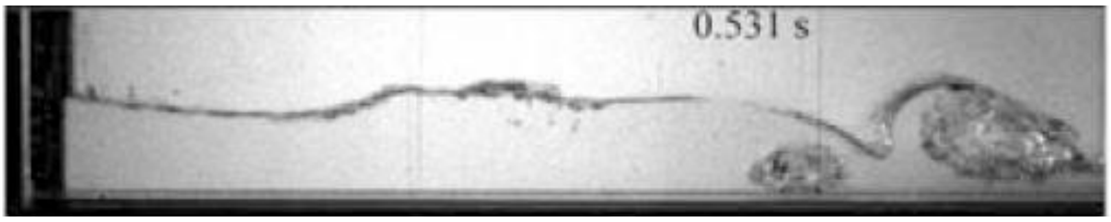
(c)



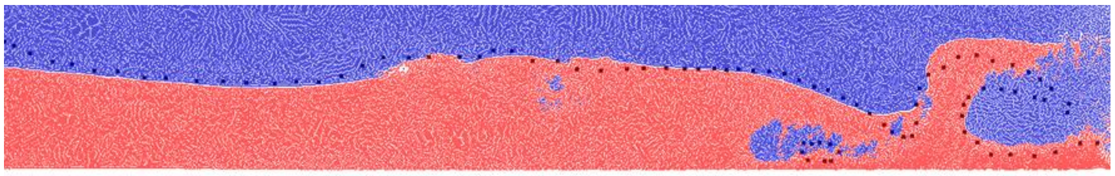
0.468s

Figure 5.41: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.468s for resolution *dx/h*$_0$=0.0067. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

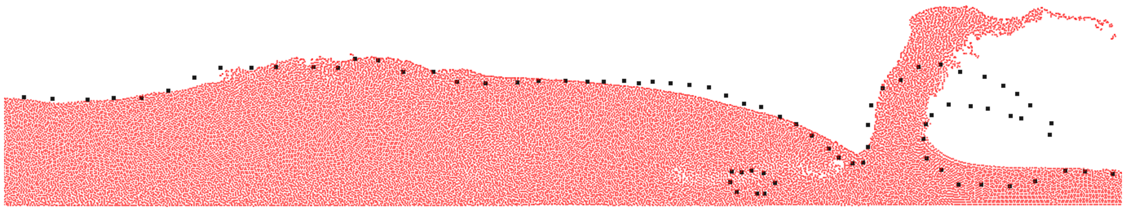
(a)



(b)



(c)



0.531s

Figure 5.42: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.531s for resolution *dx/h*$_0$=0.0067. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

5.4.4. High Water Layer: d_0=0.038m

This case is the second part of the experiments of Janosi *et al.* (2004) where the water layer height is 0.038m and the height ratio of the two water volumes is 0.2533. The speeds of sound for the different cases, as well as the artificial viscosity coefficient are the same as in the low water layer case. The particle spacing used for the first simulation is dx/h_0=0.1333. This case has been simulated by Crespo *et al.* (2008) who have digitised the screenshots provided by Janosi *et al.* (2004). The gate movement has been delayed as proposed by Violeau and Issa (2007b).

As we can see in Figure 5.43 and Figure 5.44 the experimental results are reproduced by the SPH simulations. The differences between the two simulations are very small when tracking the free surface. The backward is similarly reproduced with minor differences. An identifiable difference only occurs for 0.219s with the multi-phase model identifying the mixing between the two fluids.

Figure 5.45 and Figure 5.46 show again the phase issue observed for the low water layer case in section 5.5.3. The wave created by the forward flow breaks faster than the experimental results would indicate, especially for the multi-phase test case. Figure 5.45 also shows that the single-phase simulation predicts the breaking at the back of the wave at the correct moment. However, Figure 5.46 shows that the phenomenon is quickly eliminated without the presence of the air particles. Their presence also allows the multi-phase simulation to maintain the mixing between the two fluids inside the water flow, although in the experiment the air volume is much larger.

The latter stages of the high water layer (d_0=0.038mm) case focus on two areas: the splash up and the air bubble at the front and the mixing of air and water at the middle of the flow. The splash-up is not correctly represented in either simulation with the multi-phase splash-up adopting a mushroom shape instead of a plunging wave and the single-phase one being dispersed. The mixing between the phases is predicted by the multi-phase but like in Figure 5.46 its effect is underestimated. It cannot however be predicted by the single-phase model. The entrained air bubble is modelled by the multi-phase simulation but its volume is underestimated. The single-phase simulation, on the other hand overpredicts the volume of the bubble as well as placing it closer to the free surface.

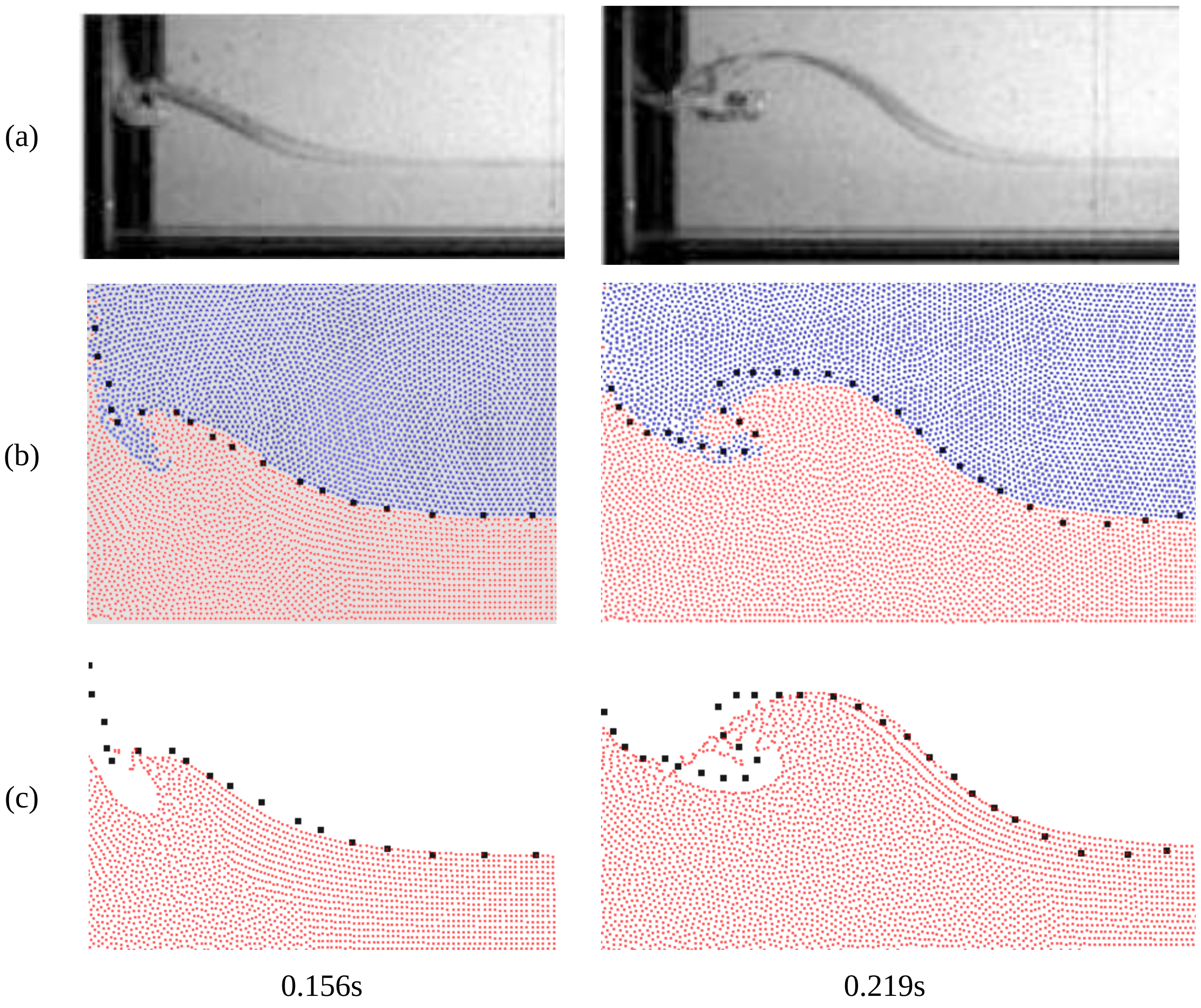


Figure 5.43: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.156s and *t*=0.219s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

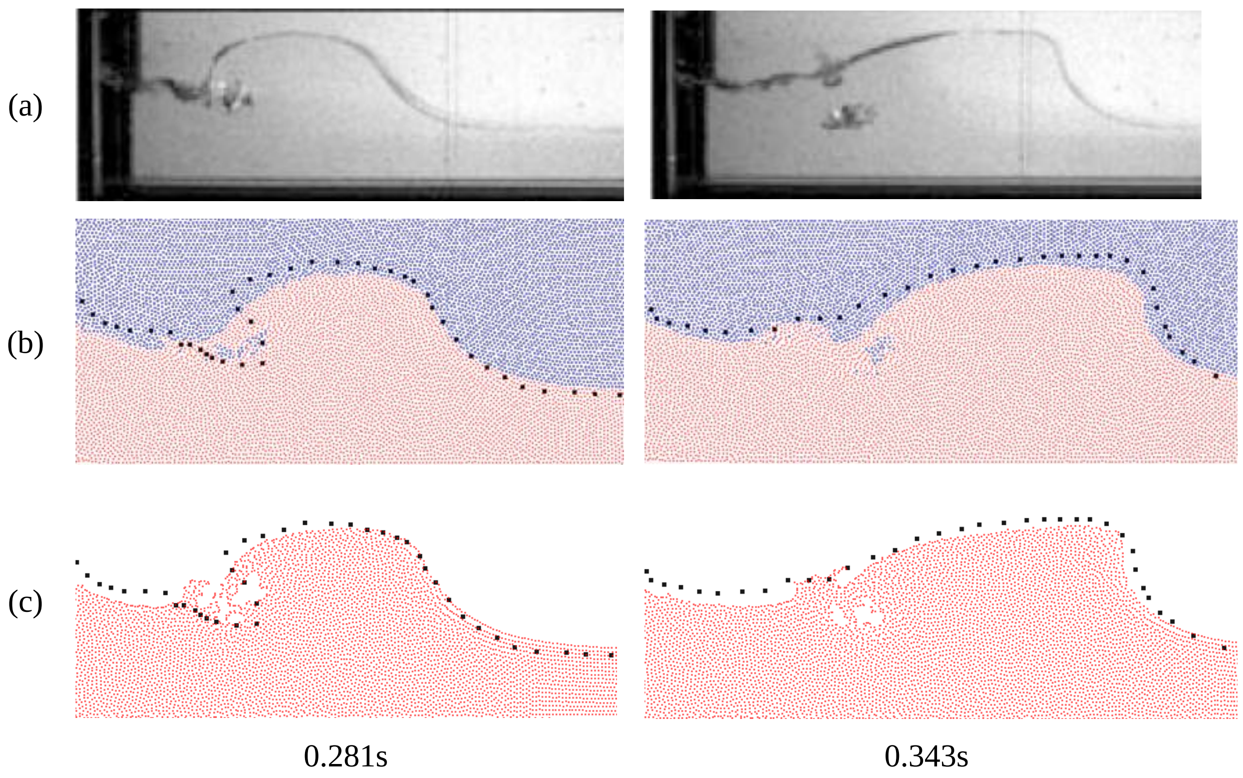


Figure 5.44: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.281s and *t*=0.343s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

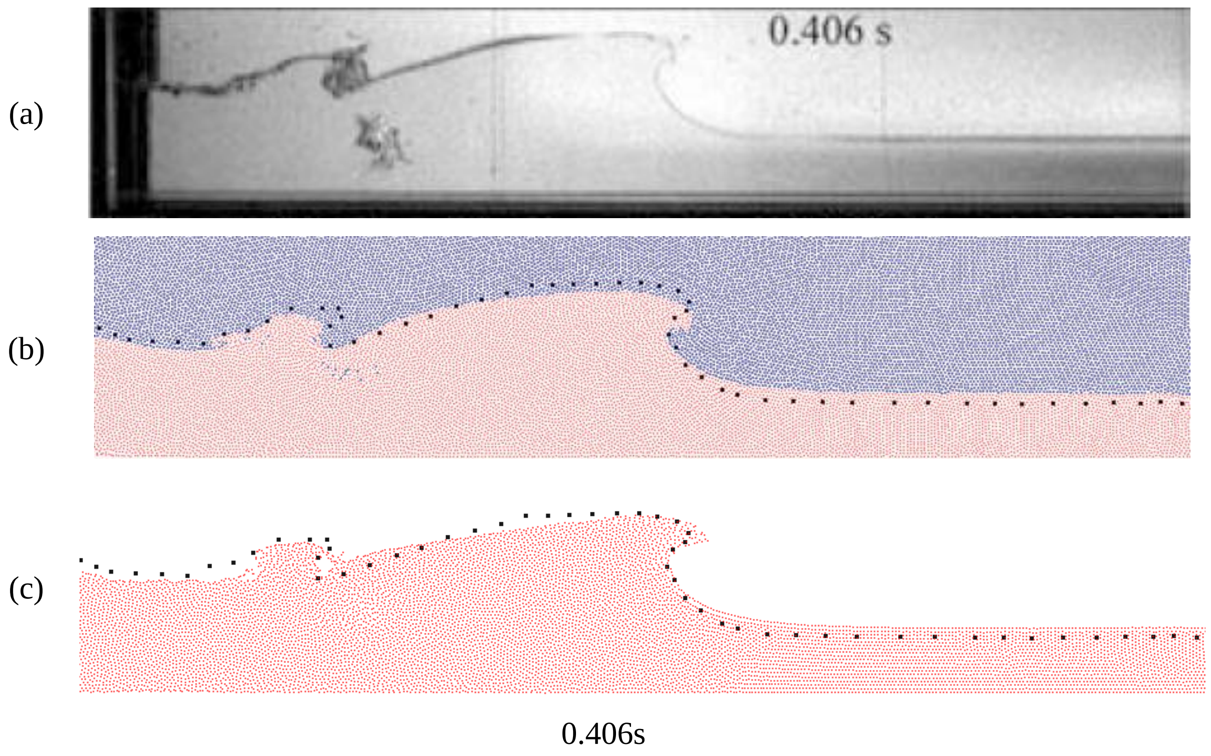


Figure 5.45: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.406s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

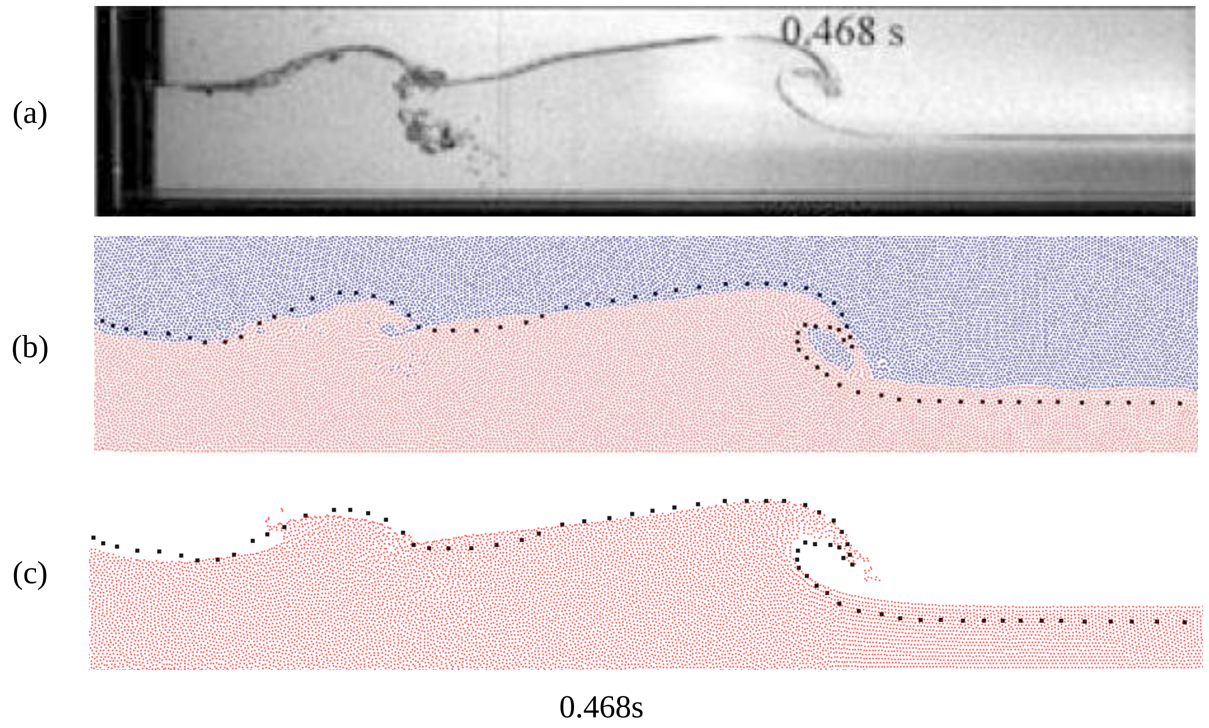


Figure 5.46: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.468s for resolution *dx/h₀*=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

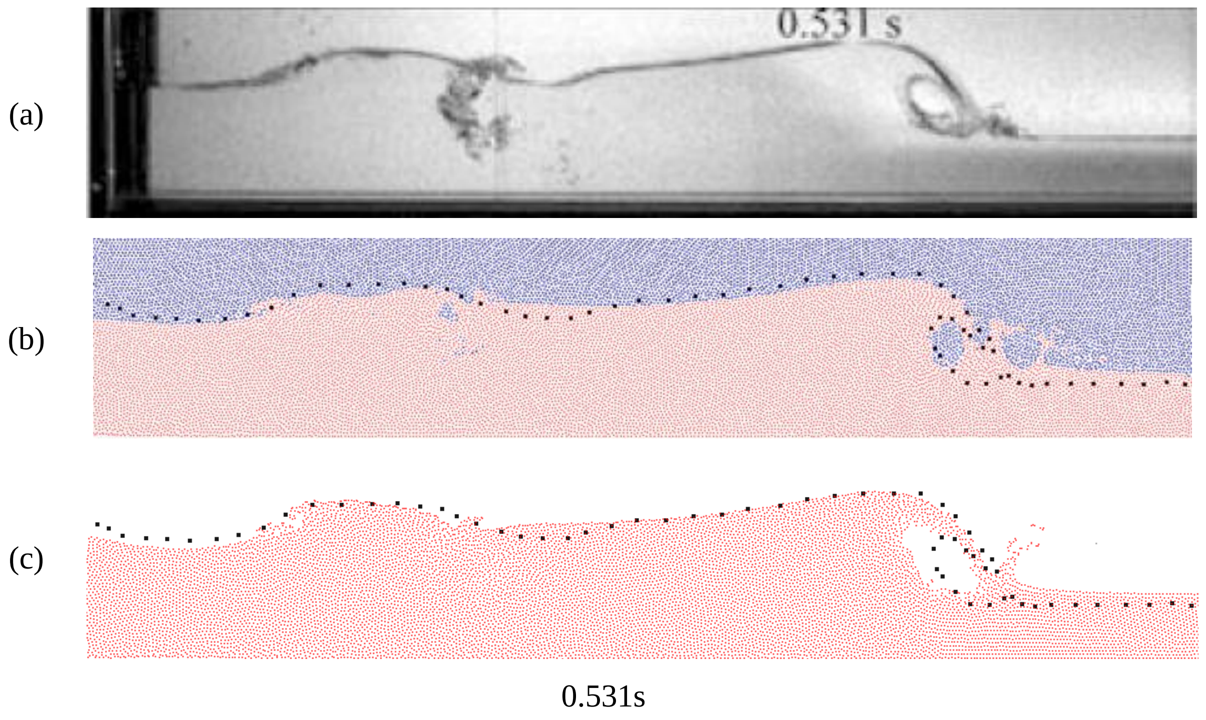


Figure 5.47: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.531s for resolution *dx/h₀*=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

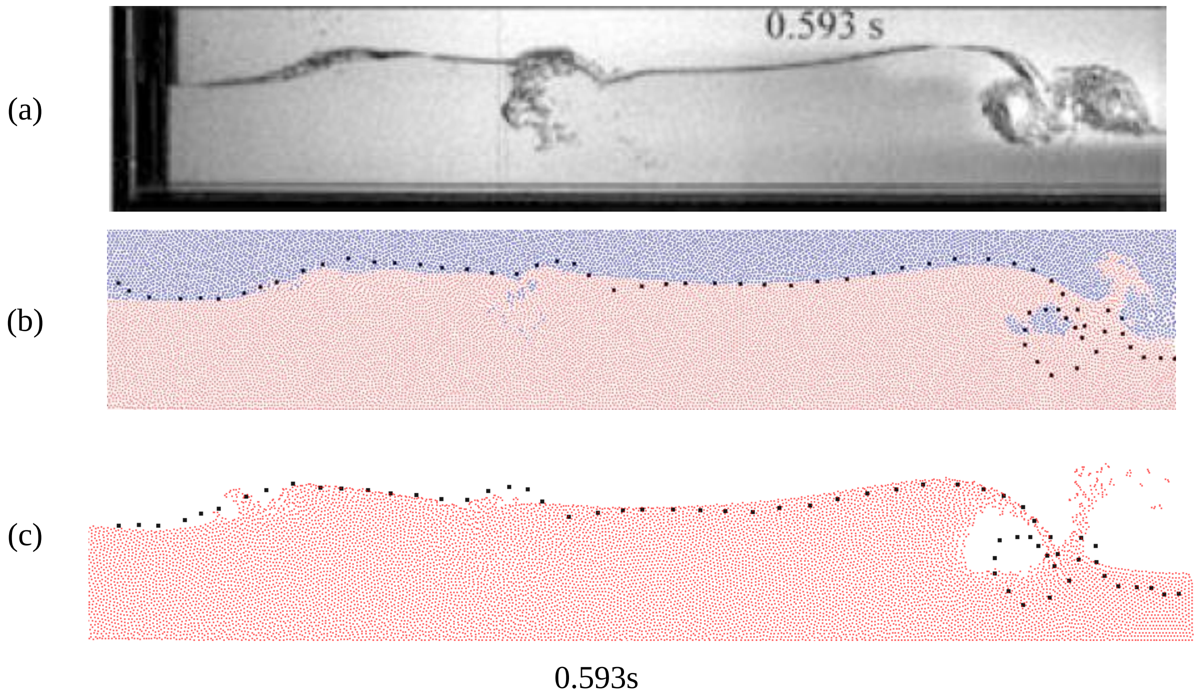


Figure 5.48: Comparison of the experimental results to a single and a multi-phase simulation for *t*=0.593s for resolution *dx/h*$_0$=0.0133. (a) Photos form experiment (b) multi-phase SPH on GPU (c) single-phase SPH on GPU. • represent experimental data

The case has also been executed for a larger resolution, with particle spacing dx/h_0=0.067, doubling the resolution of the previous simulation with the same parameters for the speeds of sound and the artificial viscosity being used. This is the same spacing used for the previous wet dam break case. The instance from the simulation occurs 0.043s earlier following the phase difference between the experimental and numerical results as outlined by Violeau and Issa (2007b).

The differences between the two models in Figure 5.49 are small, except for the volume of the backwards breaking wave, which is slightly larger for the single-phase simulation. However, the differences are more pronounced in Figure 5.50 where the evolution of the breaking wave is different for each simulation. The larger wave of the single-phase model is breaking earlier, creating a smaller space within the flow. This then evolves into a small forward collapsing wave which is not observable in the experiment.

The multi-phase simulation creates a larger air bubble that then bursts in a matter similar to the experimental results. However, the volume of the bubble is greater than the experiment and the additional entrained bubble inside the flow is not captured. An issue with both simulations is that the height on the back of the emerging wave is underestimated.

The evolution of the plunging wave can be seen in Figure 5.51 and Figure 5.52, showing the creation and the breaking of the plunging wave. Small differences between simulations can be seen at the tip, with the wave of the multi-phase simulation being forced to a quicker breaking. Of interest is the evolution of the smaller wave breaking that occurs at the back of the flow; the single-phase simulation creates an upwards splash-up that does not occur in the experiment, while the multi-phase model shows a more realistic breaking.

The final two images, Figure 5.53 and Figure 5.54, show the resulting splash-up from the plunging wave. Similar to the low water layer presented in Section 5.5.3 the height of the splash-up is overestimated and the volume of the entrained air is better simulated with the air particles. Air-water mixing within the flow can be seen at the multi-phase simulation but the voids inside the single-phase model have vanished. The voids' presence within the single-phase flow was only recorded because the time difference between the figures is very small; without the presence of another phase, they cannot be maintained.

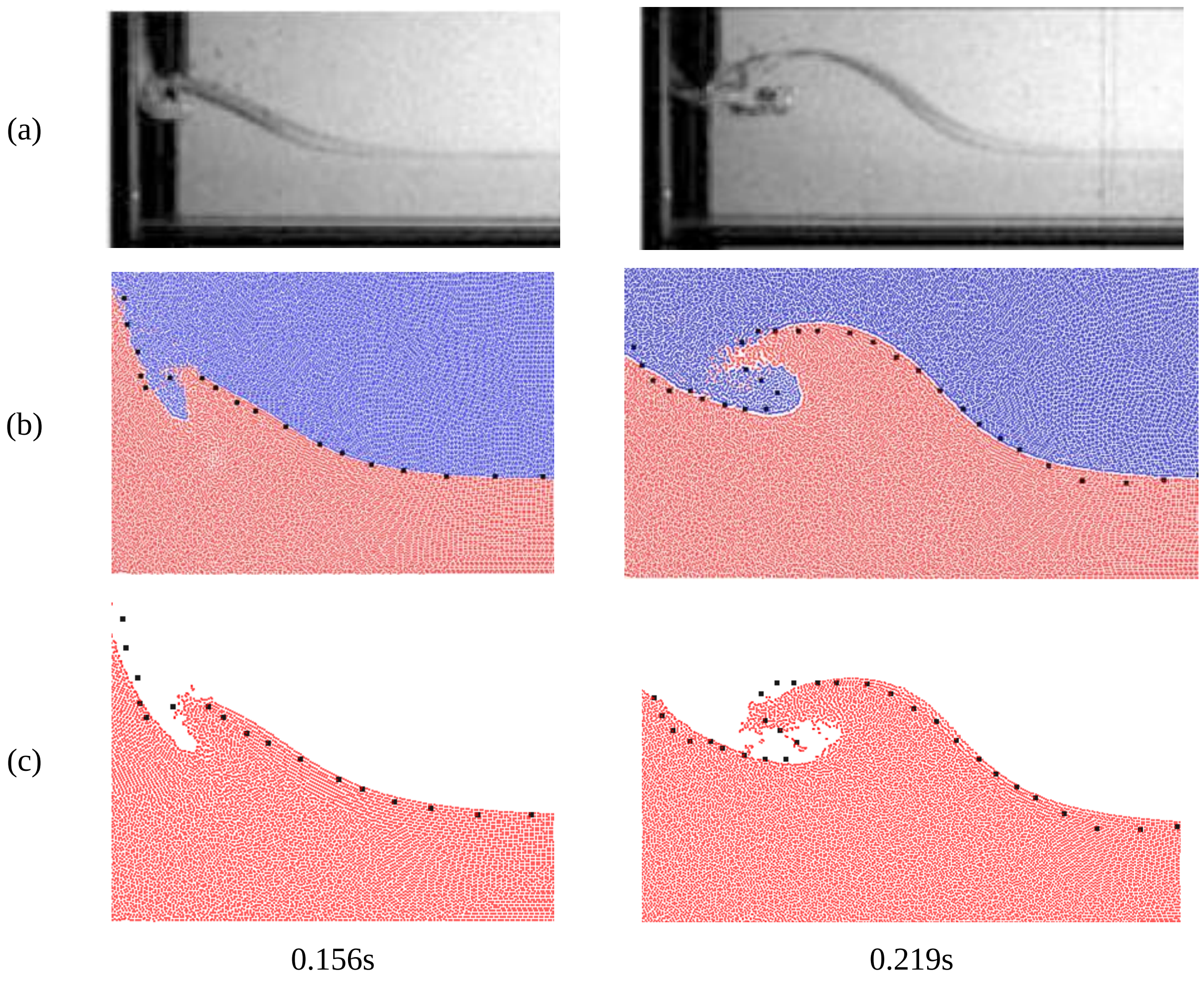


Figure 5.49: Comparison of the experimental results to a single and a multi-phase simulation at the start of the computation for dx/h_0=0.0067

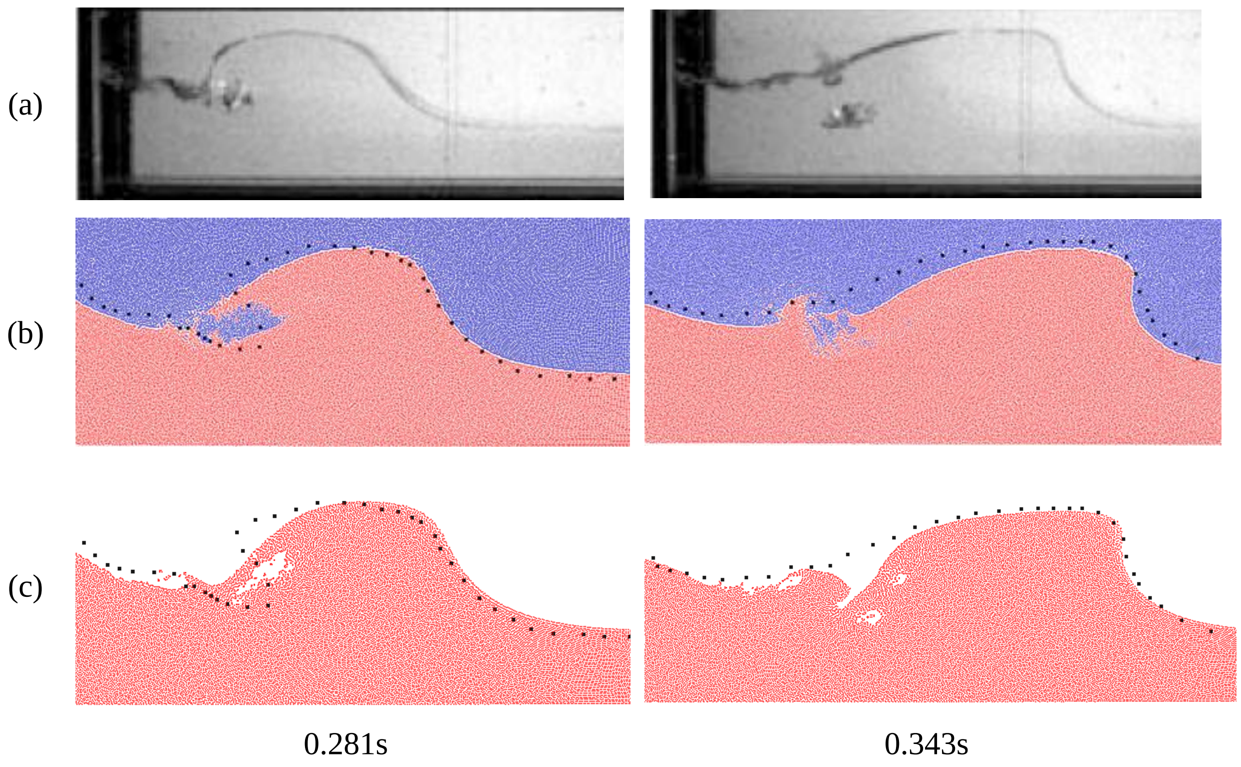


Figure 5.50: Comparison of the experimental results to a single and a multi-phase simulation at 0.281s and 0.343s for dx/h_0=0.0067

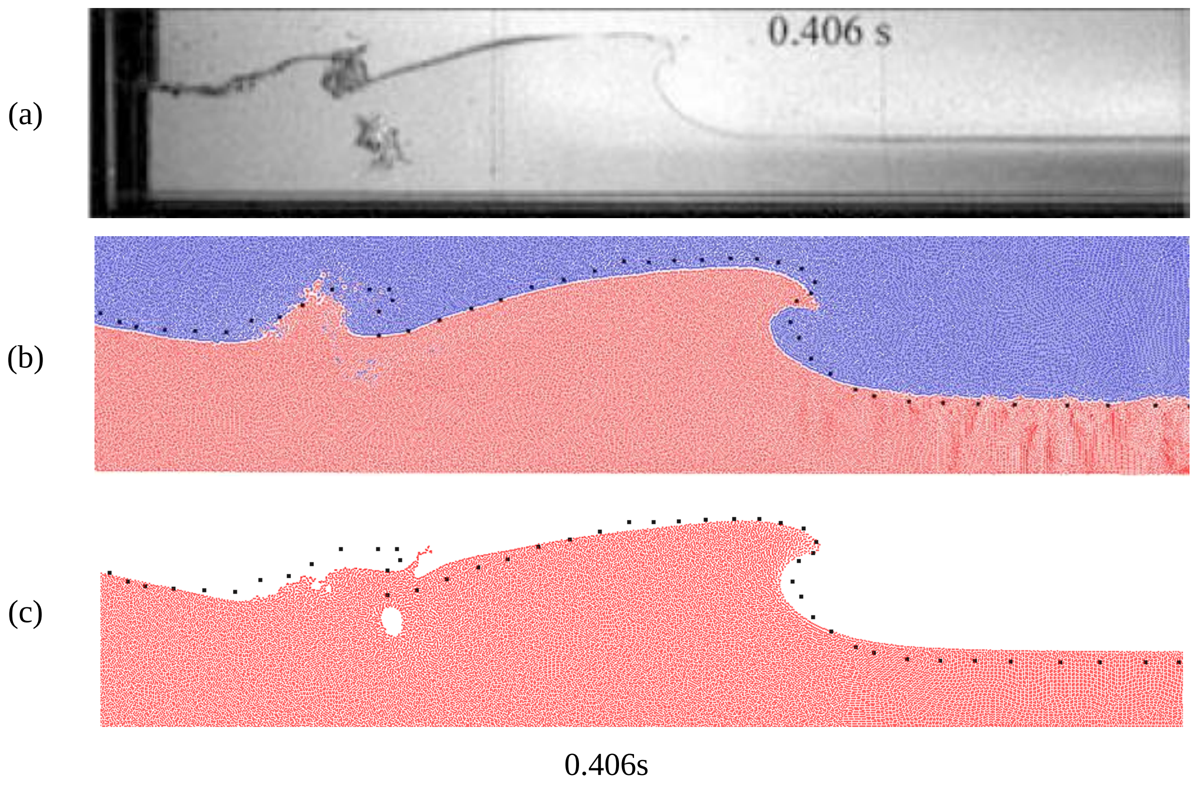


Figure 5.51: Comparison of the experimental results to a single and a multi-phase simulation at 0.406s for dx/h_0=0.0067

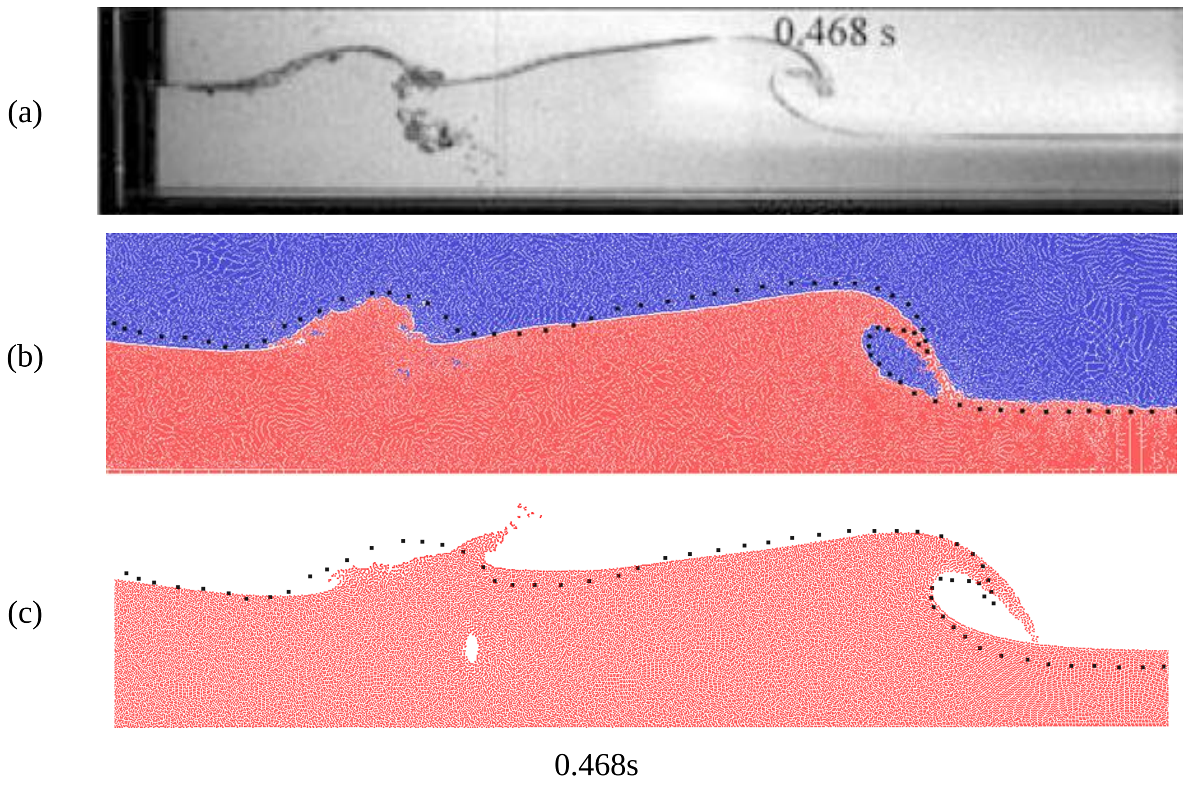


Figure 5.52: Comparison of the experimental results to a single and a multi-phase simulation at 0.468s for *dx/h*$_0$=0.0067

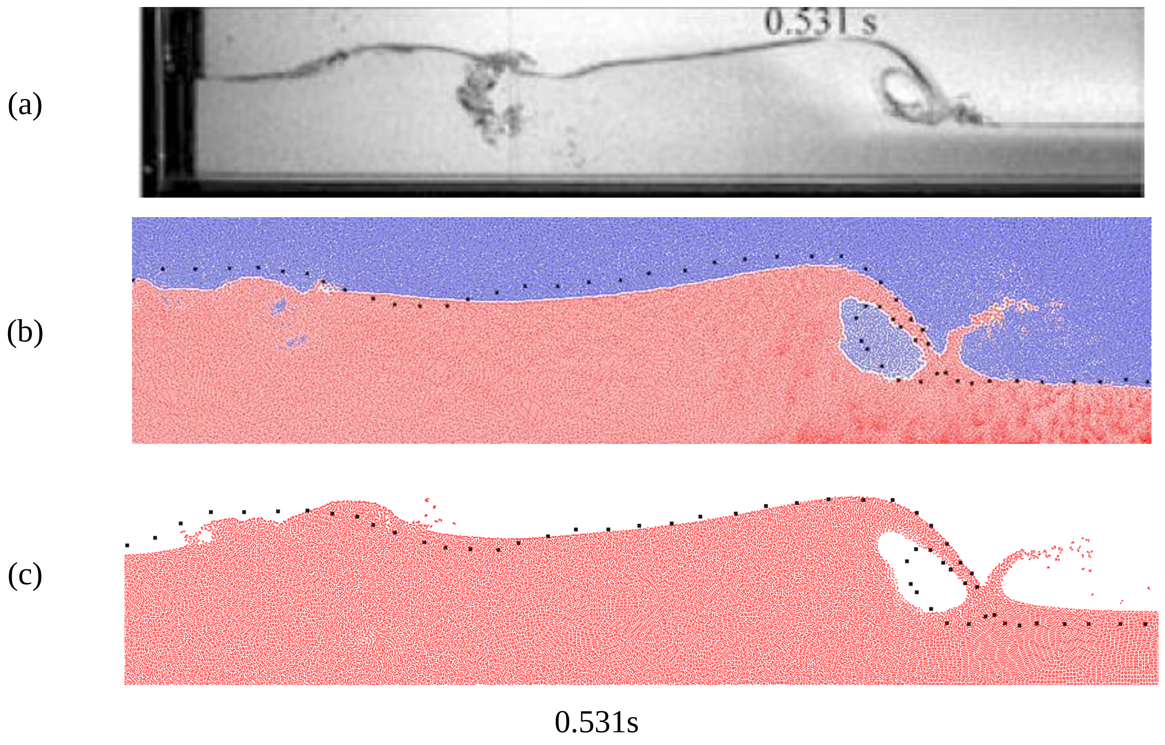


Figure 5.53: Comparison of the experimental results to a single and a multi-phase simulation at 0.531s for *dx/h*$_0$=0.0067

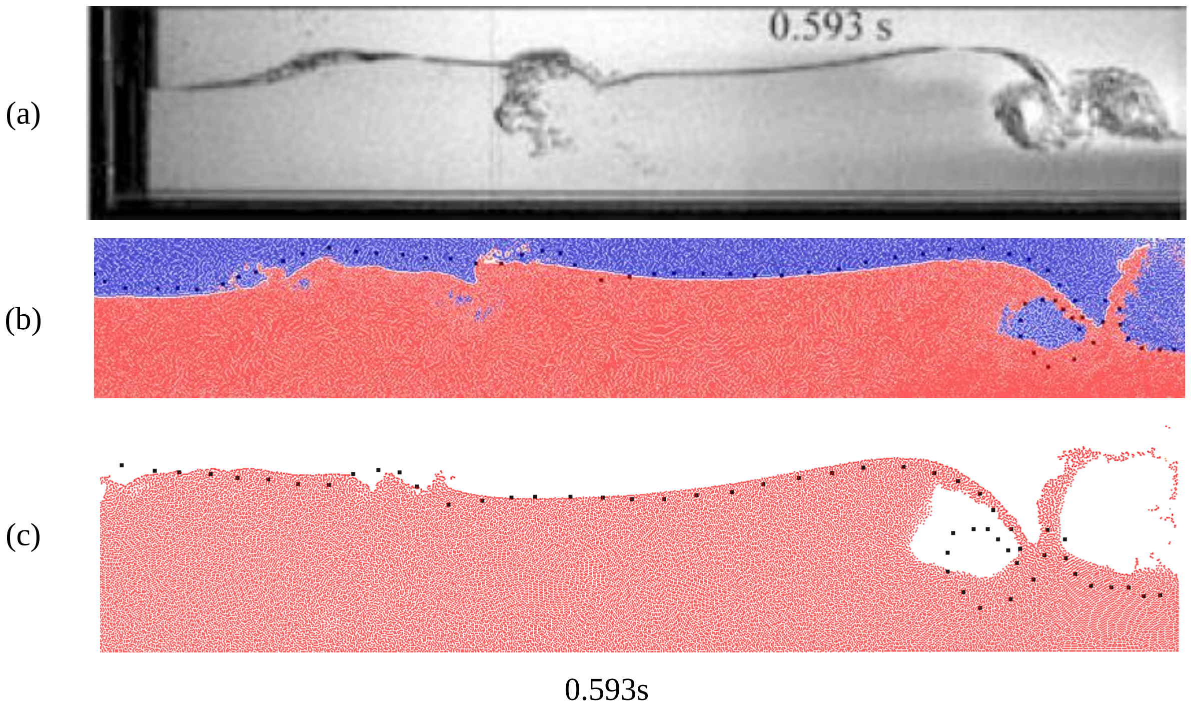


Figure 5.54: Comparison of the experimental results to a single and a multi-phase simulation at 0.593s for *dx/h*$_0$=0.0067

Compared to the low water layer (d_0=0.018m) presented in Section 5.4.3, differences between the high and low resolutions are more pronounced in this case, especially for the single-phase test case. The front of the flow does not differ with the resolution but the backwards breaking and its evolution exhibit different behaviour. For the single-phase the results seem to be better for the lower resolution, indicating that there are issues with the viscosity model in the higher resolutions.

Although not reported here, numerical comparison of multi-phase and single-phase SPH schemes with shifting for the initial stages of a wet-bed dam break flow show good agreement with a semi-analytical solution (Stansby *et al.*, 1998), indicating that treatment of the gate removal affects the results as confirmed by Violeau and Issa (2007b).

5.5. Rolling Tank

5.5.1. Case Description

SPHERIC Benchmark Case 10 (Botia-Vera *et al.*, 2010, Souto-Iglesias *et al.*, 2011a) was selected to experimentally validate the multi-phase SPH code used in this study. The experiment studies the movement and the wall forces at a sloshing tank case. The case was selected in order to measure the accuracy of the pressure field of the simulation to the experimental results, as well as demonstrate the effectiveness of the multi-phase model for a case with a very sensitive interface.

The importance of this case lies in measuring the forces exerted on the walls of the tank. Depending on the case and the volume of the tank and the liquid the pressure rise inside the vessel can be quite high and the subsequent forces can severely damage the walls or the support structure. The forces are particularly high when there is resonance between the liquid movement and the craft motion. The changes in the weight distribution can also have a negative impact on the support structure of the tank.

For a rectangular tank, the sloshing phenomena can be approximated using a two-dimensional simulation. If forces on the fluid caused by the vessel movement change directions frequently, the flow near the corners becomes three-dimensional. For spherical or cylindrical tanks, on the other hand the three-dimensional effects have to be taken in consideration, especially in regards to the support structure (Pauling, 2008).

Liquid sloshing in rectangular tanks can be divided in two categories: low and high-filling tanks. In the first case, the sloshing effect is characterised by travelling waves and hydraulic jumps. Near the resonance frequency this can lead to high pressure impacts on the walls. A typical pressure distribution on the tank wall for a low-filling case can be seen in Figure 5.55. The high-filling cases on the other hand are characterised by large standing waves and the main pressure impacts are on the roof of the vessel instead of the walls.

The most disastrous effect caused by the liquid sloshing is the capsizing of the craft containing the tank. The movement of the liquid can change the centre of mass for the vessel resulting in it becoming unstable and rolling over. This is a grave problem in craft with large fluid cargo, especially if it is spanning the full breadth of the craft. In a ship, accidental flooding of its compartments can potentially have the same effect. The effect is greater with a partially-filled tank, due to the larger distances the liquid can move.

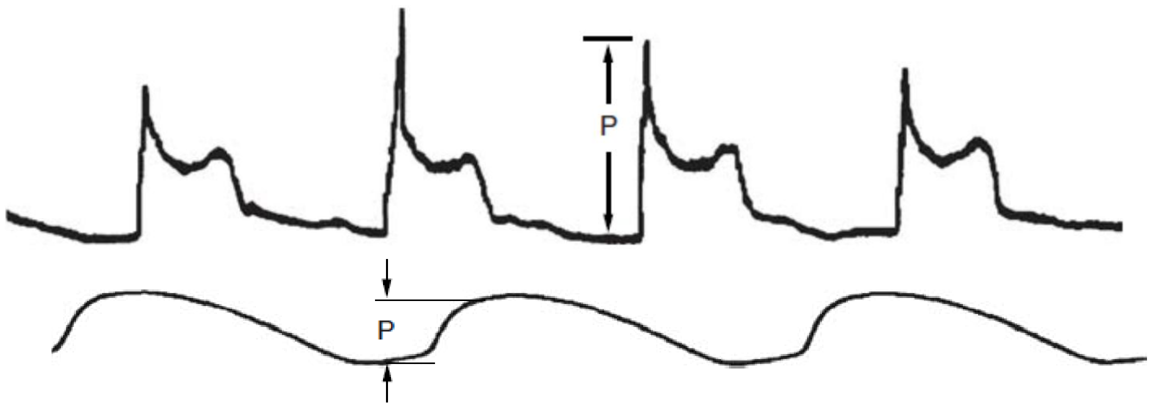


Figure 5.55:Pressure time history for a) resonant sloshing with large amplitude hydraulic jumps and b) non-resonant low amplitude standing wave sloshing (Bass *et al.*, 1976)

The sloshing effects usually result in a violent flow with rapid changes of the free surface. The tanks are only partially filled so the liquid flow is mixing with the air or other gases that are inside the tank. The waves created inside the tank have also a high possibility of trapping gas volumes within the liquid flow. This occurs regardless of the water depth in the tank. The forces on the walls are, as a result, affected.

In this test case the lateral impact in a rectangular tank is considered. The liquid height used corresponds to 18% filling tank level. The level was selected in order to maximise the wall impact. Experimental data are available for the first four impacts with the pressure at certain points on the walls being recorded as well as the roll angle of the tank. Data were recorded for two fluids, water and sunflower oil, but only the water results are considered.

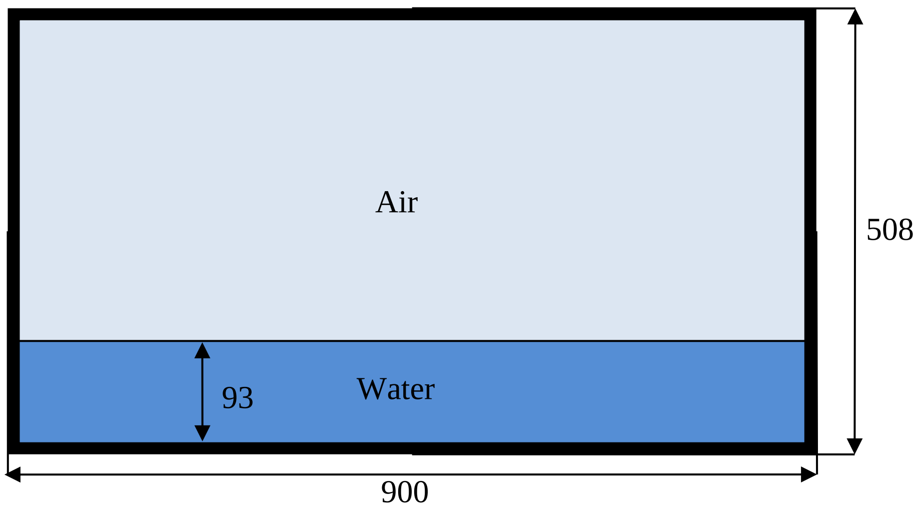


Figure 5.56: Definition sketch for the lateral impact of SPHERIC Benchmark Case 10

The movement of the liquid inside depends on the tank movement. In this case the tank is rotated along over a steady point at the middle of its base. The maximum rotation angle is 4 degrees. The pressure on the walls is measured in the positions shown in Figure 5.57. Of primary interest for this case are the positions on the tank walls near Sensor 1.

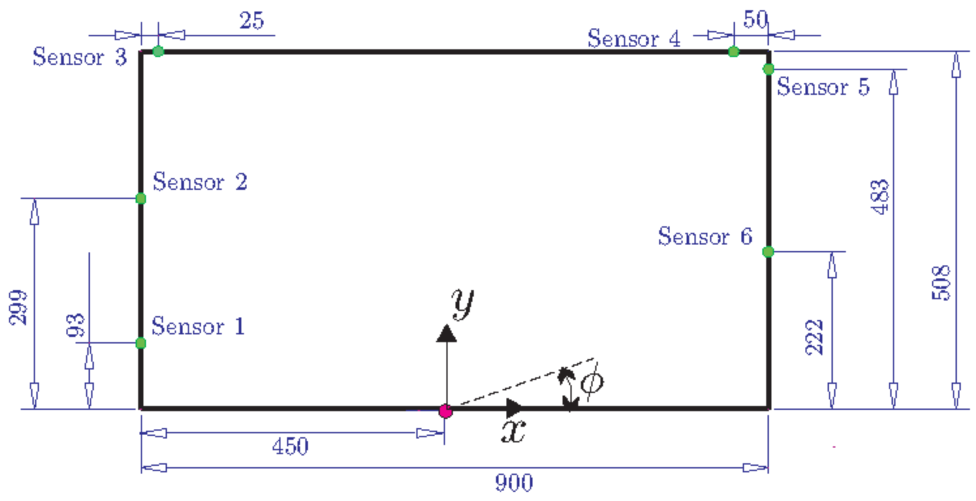


Figure 5.57: Tank Geometry and pressure sensor position of the sloshing flow tank (Souto-Iglesias *et al.*, 2011b)

In the simulation the boundary particles simulating the tank walls remain stationary for the duration of the simulation. The movement of the tank is instead simulated by changing the gravity vector in accordance with the roll angles of the experiment. The angular acceleration is also being computed. The gravity force that is being applied in the horizontal axis is shown in Figure 5.58.

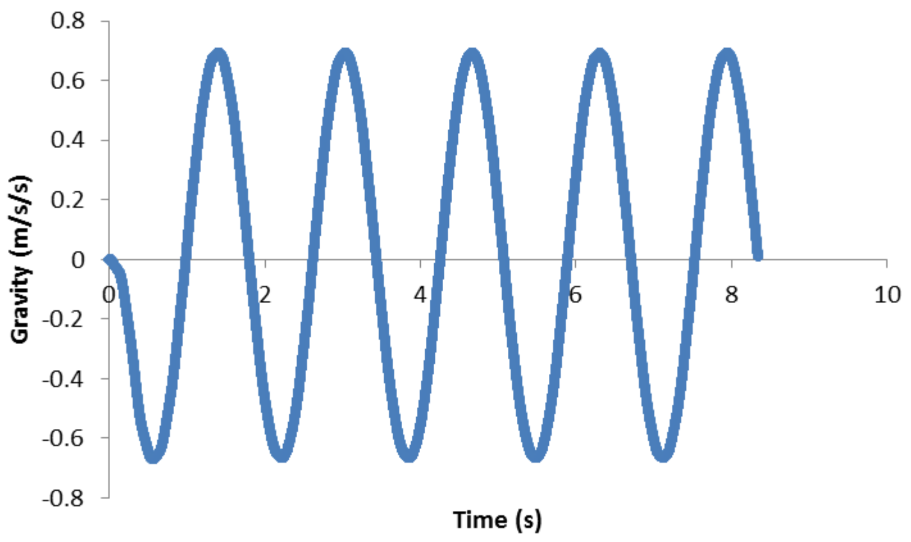


Figure 5.58: Gravity Force applied on the horizontal axis

The pressure profiles for these cases have been measured for sensors 1 for the lateral impact case and 3 for the roof impact case and can be seen in Figure 5.59.

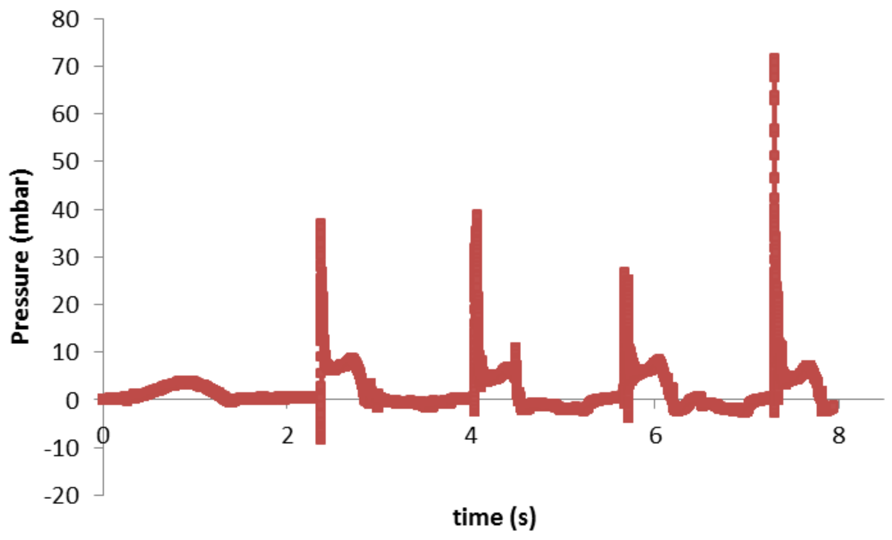


Figure 5.59: Expected Pressure for sensor 1

5.5.2. Lateral impact test case

The lateral impact case has been simulated with SPH by Leonardi *et al.* (2011) who used the artificial viscosity scheme, running a sensitivity analysis on the viscosity parameter *a*. For the velocity and the pressure they used an adopted XSPH correction scheme. The eddy viscosity is also calculated, either with a *k*-*ε* model or with a mixing length model (De Padova *et al.*, 2010).

Leonardi *et al.* (2011) found that the lateral impact case was very sensitive to the viscosity of the flow. The pressure peaks in particular, would vary significantly depending on the viscosity model used and the value of its coefficients. The profile of the water flow also depends on these models, with some configurations creating breaking waves and others maintaining a smooth interface for the whole duration of the simulation.

In this case, this study uses neither a turbulence model, nor the XSPH scheme. Instead, the shifting algorithm for both phases is being used, with the water phase being treated with the free surface term as well as a Shepard renormalisation scheme. The equation of state used is also different. Regarding these differences, the effective viscosity field of the two simulations is sufficiently different to make a direct comparison impossible. This study will instead focus on the differences seen in the pressure field, as recorded in the area of sensor 1 for different parameters in this study.

An interesting aspect of the case is how the different phases impact the pressure. Due to its oscillatory nature and the position of the sensor, the influence of the two phases alternates. It is easy then to find at which times the pressure are dictated by each phase. The profile can then be seen in Figure 5.60 and it is observed that particles from both phases contribute to the pressure peak.

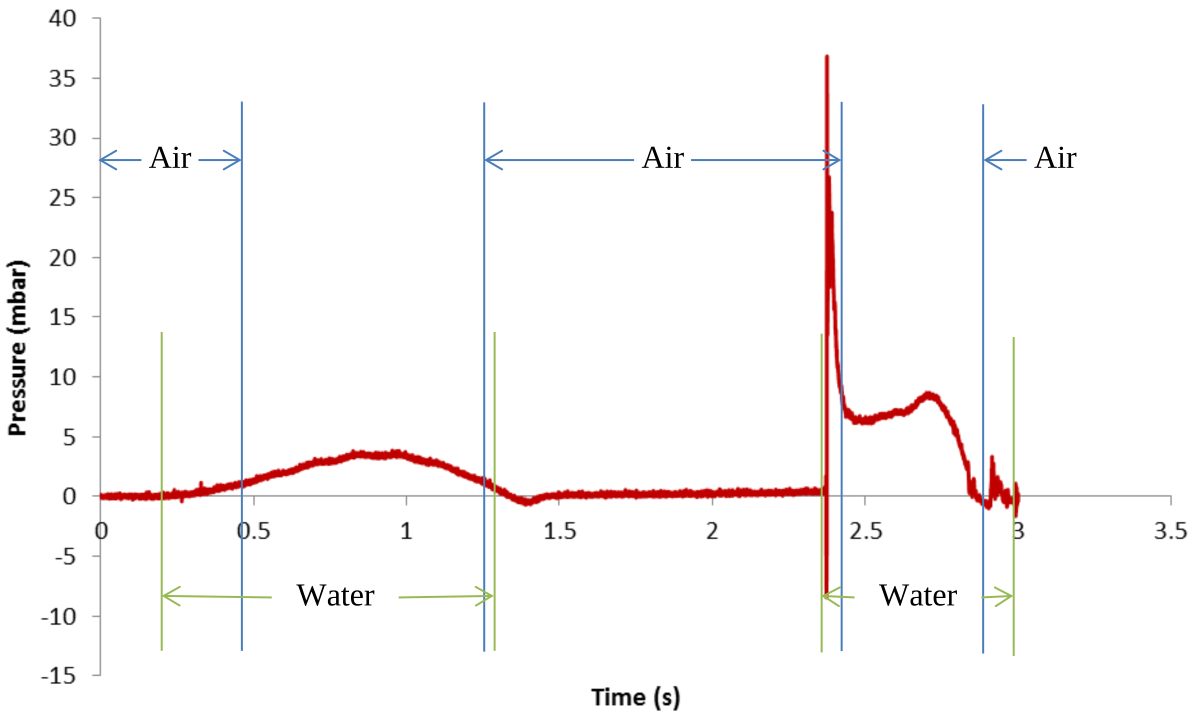


Figure 5.60: Mapping of impact on pressure by the different phases

The pressure is calculated using the MeasureTool executable provided with DualSPHysics code which can perform an interpolation at any given point in the domain given the results by DualSPHysics. The measurement probe was initially situated at the exact point that sensor 1 was situated, among the boundary particles on the wall. However, preliminary measurements showed that pressure in the boundary particle would vary rapidly depending the phase of its neighbours, resulting in an elevated, unnatural pressure that did not reflect the state of the domain.

In that regard, the probe was placed at a horizontal distance 2*h* from the boundaries, inside the fluid domain, the lowest possible distance close to the boundary while not taking into account the boundary particles and maintaining a full kernel. The distance from the wall is small enough that the pressure peak can be captured without the smaller pressure values inside the flow impacting the end result.

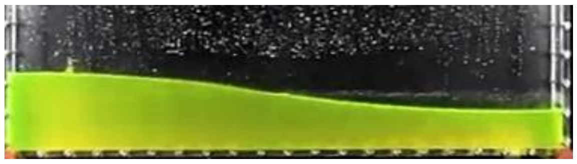
Due to the presence of two fluids, direct recording of the pressure using MeasureTool was impossible. Instead, two cases were recognised; whether the neighbouring particles of the probe were of a single or of both phases. If only particles of a single phase were present a density interpolation was performed using MeasureTool and the pressure was calculated using the equation of state. If both air and water particles were present, the particle values were extracted using Paraview and an interpolation was performed manually.

Several multi-phase simulations have been conducted. Figure 5.61 to Figure 5.65 show instances for two resolutions with a particle spacing of dx/h_0=0.0215 and dx/h_0=0.0161, respectively, where h_0 is the initial height of the water in the tank has been executed and dx is the particle spacing. Regarding the speeds of sound, $c_{s,a}$ is the speed of sound for the air phase, while $c_{s,w}$ is the equivalent value for the water phase. They are compared to the experimental results of Botia-Vera *et al.* (2010). The instances selected were at 1s, where the pressure is first increased, at 2s, at 2.37s and 2.45s, right before and while the first impact occurs and at 3s.

Agreement with the experimental screenshots is generally good, but there are some differences to be found in the later stages of the simulation. Specifically, in Figure 5.63, the simulation depicts a plunging wave, while in the experimental screenshot the wave has already collapsed and has initiated the impact with the wall. This occurs for both resolutions, with the lower one being slightly delayed.

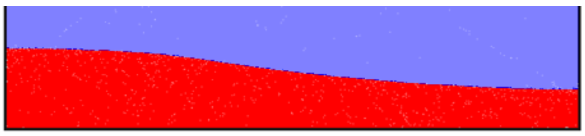
A similar situation can be found in Figure 5.65 where a breaking wave starts occurring for the lower resolution while the experiment and the higher resolution simulation retain a smooth free-surface. The higher resolution has a slightly better agreement with the experimental values (Figure 5.62 and Figure 5.65), but has also some boundary issues, similar to the dry dam break case in Section 5.2, more apparent in the right side of Figure 5.62.

Experiment Photo



Multi-phase SPH simulation with

dx/h_0=0.0215



Multi-phase SPH simulation with

dx/h_0=0.0161

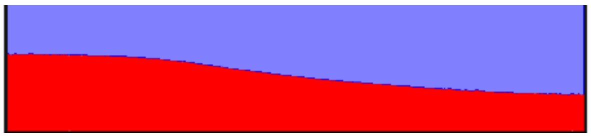
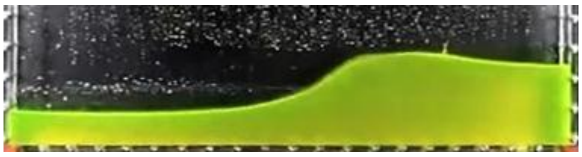


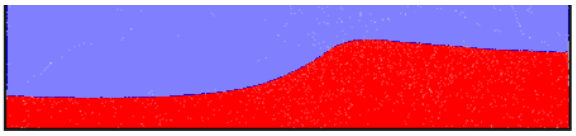
Figure 5.61: Comparison between an experiment screenshot and multi-phase simulation with $c_{s,w}$=20m/s, $c_{s,a}/c_{s,w}$=7.5, α=0.01 instances at 1s

Experiment Photo



Multi-phase SPH simulation with

dx/h_0=0.0215



Multi-phase SPH simulation with

dx/h_0=0.0161

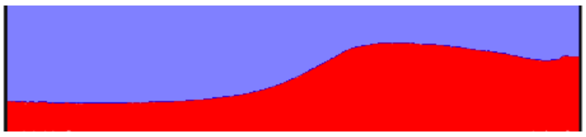
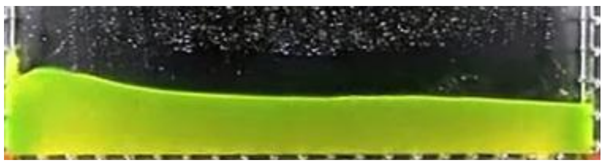


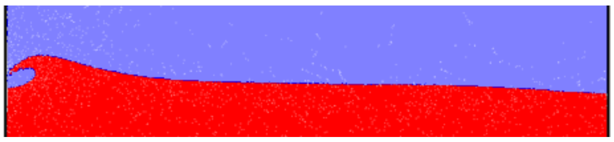
Figure 5.62: Comparison between an experiment screenshot and multi-phase simulation with $c_{s,w}$=20m/s, $c_{s,a}/c_{s,w}$=7.5, α=0.01 instances at 2s

Experiment Photo



Multi-phase SPH simulation with

dx/h_0=0.0215



Multi-phase SPH simulation with

dx/h_0=0.0161

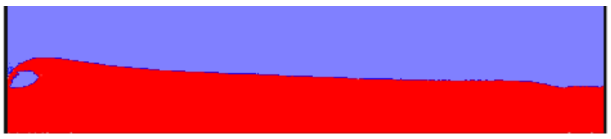
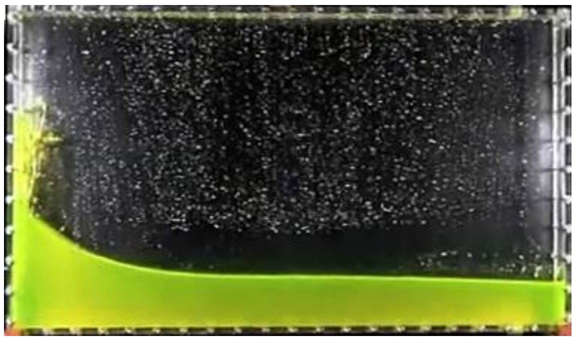


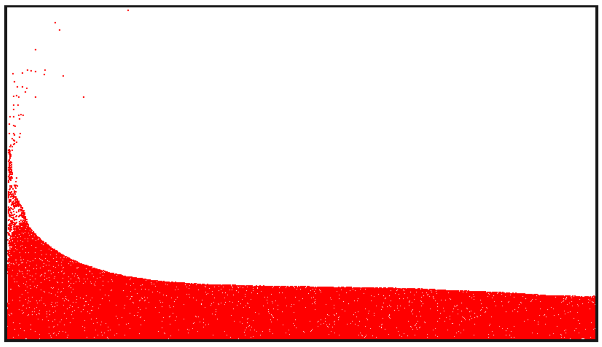
Figure 5.63: Comparison between an experiment screenshot and multi-phase simulations with $c_{s,w}$=20m/s, $c_{s,a}/c_{s,w}$=7.5, α=0.01 instances at 2.37s

Experiment Photo



Multi-phase SPH simulation with

dx/h_0=0.0215



Multi-phase SPH simulation with

dx/h_0=0.0161

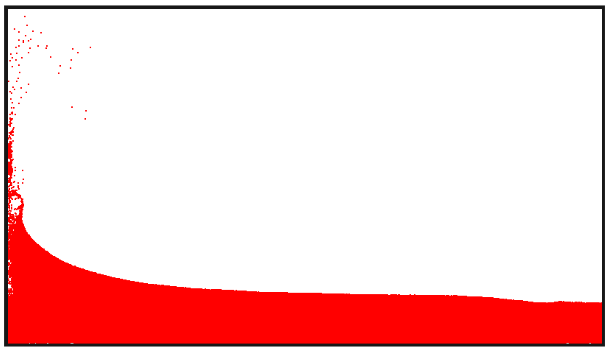
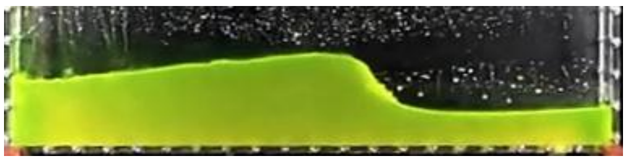


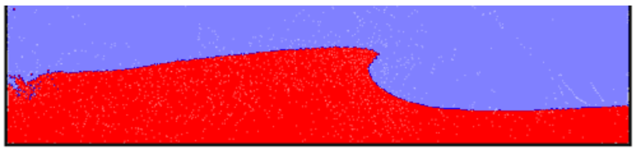
Figure 5.64: Comparison between an experiment screenshot and multi-phase simulations with $c_{s,w}$=20m/s, $c_{s,a}/c_{s,w}$=7.5, α=0.01 instances at 2.45s. The air phase has been omitted for clarity

Experiment Photo



Multi-phase SPH simulation with

dx/h_0=0.0215



Multi-phase SPH simulation with

dx/h_0=0.0161

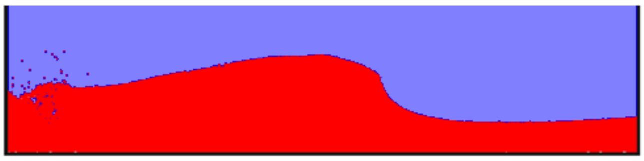


Figure 5.65: Comparison between an experiment screenshot and multi-phase simulations with $c_{s,w}$=20m/s, $c_{s,a}/c_{s,w}$=7.5, α=0.01 instances at 3s

The simulation performed with dx/d_0=0.0215 was executed with different parameters. The value proposed by Leonardi *et al.* (2011) for the speed of sound of the water was 17m/s and it was found with preliminary simulations that choosing the speed of sound for the water to be in the area around 20m/s gave the best fit with the experimental results. The first results are presented in Figure 5.66.

The speed of sound for the air was selected in accordance with the method proposed by Colagrossi and Landrini (2003) for ensuring that the pressure at the interface would be identical for both phases if they had their initial density. The ratio for an air and water flow is 29 and it is the value used in this case. The value for the artificial viscosity coefficient is 0.01, the value proposed by Leonardi *et al.* (2011).

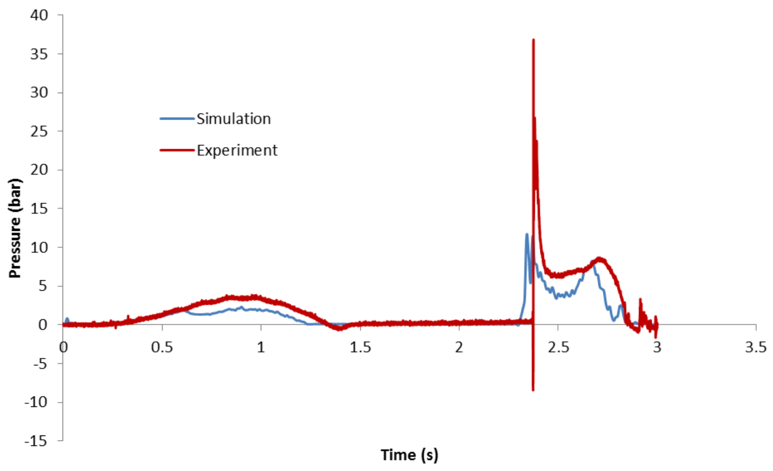


Figure 5.66: Comparison between experimental and simulation results for $c_{s,w}$=17m/s and $c_{s,a}/c_{s,w}$=29 for dx/h_0=0.0215

As can be seen from Figure 5.66 the agreement with the experimental results is not very good. The impact occurs earlier than expected and the pressure field is much lower than the experimental results, particularly in the areas where the water has an important role. This issue is cause by the very high compressibility of the air phase, which reduces the range of values of the density resulting in the pressure peak being underestimated.

To counter this issue the speed of sound for the air was reduced. Two simulations were performed with speed of sound ratio 10.6 and 8.8. The results are displayed in Figure 5.67. Reducing the speed of sound for the air phase offers a significant improvement on the results, especially if the speed of sound ratio is around 10. Further reduction however results in an overall pressure drop for the air phase as evidenced by the pressure values before 0.5s and between 1.5s and the pressure impact. The pressure drop in the air has a profound effect on the pressure peak resulting in a value about half of the experimental.

To confirm these results the speed of sound value for the water phase was increased by 15% and executed for 3 different speed of sound ratios: 15, 10 and 7.5. The results are presented in Figure 5.68 and they are similar to the ones shown in Figure 5.67; a speed of sound ratio around 10 achieves the closest results to the experiment. Results for the higher ratio, simulation (a), after the pressure peak are very oscillatory and do not closely follow the experimental results while the results for ratio 7.5, simulation (c), underestimate the pressure peak, but are close to the experiment.

An interesting comparison can be performed between simulation (b) in Figure 5.67 and simulation (c) in Figure 5.68. The only difference between these simulations is the speed of sound of the water and the results produced are, as expected, slightly different for the areas that the water is the primary influence on the pressure. However, the results produced for the pressure peak are nearly identical, showing that the trapped air and its compressibility is the primary factor for the pressure peak value.

Regarding the effect of the artificial viscosity, Figure 5.69 shows a comparison between simulations with identical speeds of sound but with the artificial viscosity coefficient varying. The results show that the coefficient value has a great effect on the pressure peak. The optimal value is 0.01 as suggested by Leonardi *et al.* (2011) and the pressure peak value is dropping if the viscosity value is lowered. In the rest of the computation, the only effect of the viscosity is the slight raise of the pressure values as the simulation progresses; this is visible after 1.5s.

The simulations investigated so far have been performed for a single resolution, however, the wet dam break case investigated in Section 5.5 shows that the numerical viscosity depends on the resolution of the case and it is inversely proportional. Figure 5.70 shows the results for simulations with different resolutions but using the same speeds of sound and the same artificial viscosity coefficient.

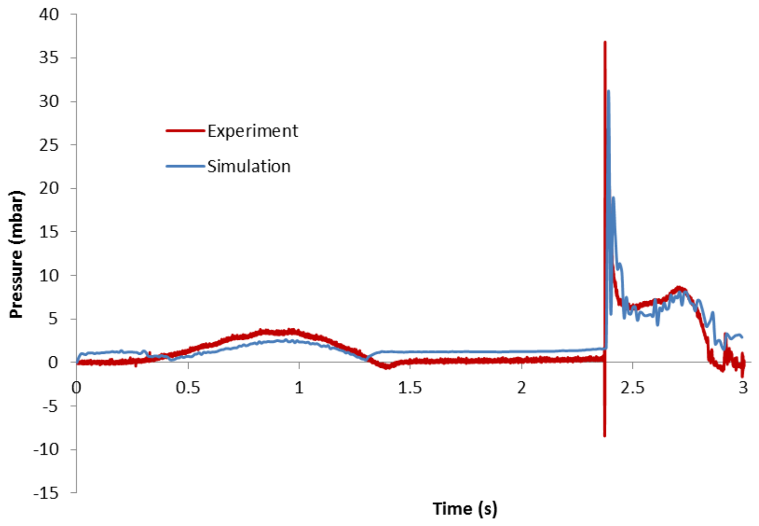
The results show that increasing the resolution leads initially to a decrease in the pressure field, especially in the areas that are heavily influenced by the water as seen in simulation (b) in Figure 5.70. If the resolution is further increased, the low speed of sound of the water leads to a spurious pressure field especially after the impact and in the underestimation of the pressure peak. The analysis performed so far however, does apply to other resolutions, with Figure 5.71 showing the best fit for the experimental results with dx/h_0=0.0161. For the higher resolution, the pressure drop at 2.35s is also captured.

This case shows that the multi-phase model used in this study is able to correctly predict the pressure field, but it also showcases its limitations. Namely, it is entirely too dependent on numerical parameters and for sensitive cases like this, achieving close results to experimental studies requires rigorous sensitivity analysis, the results of which are also dependent on the particle resolution. Regardless, it is still possible to achieve a result close to the experimental values.

a) $c_{s,w}$=17m/s

$c_{s,a}/c_{s,w}$=10.6

α=0.01



b) $c_{s,w}$=17m/s

$c_{s,a}/c_{s,w}$=8.8

α=0.01

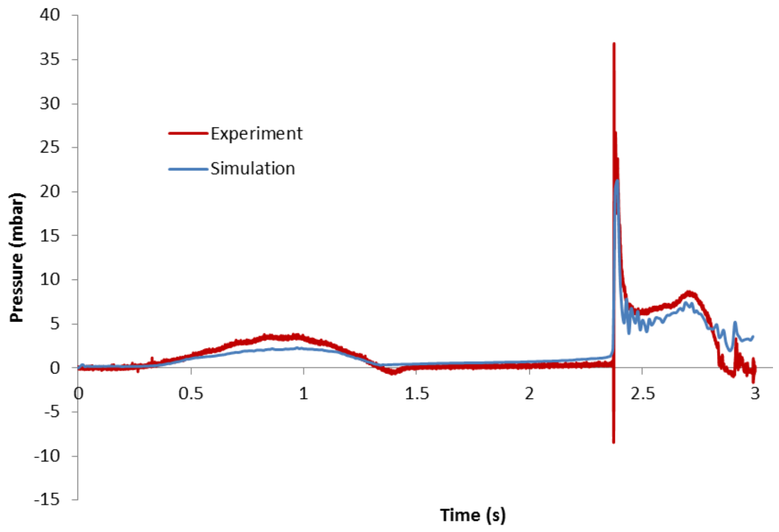


Figure 5.67: Comparison between experimental and simulation results for $c_{s,w}$=17m/s and for *dx/h_0*=0.0215. Investigation of the effect of the speed of sound for the air phase

a) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=15

α=0.01

b) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=10

α=0.01

c) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=7.5

α=0.01

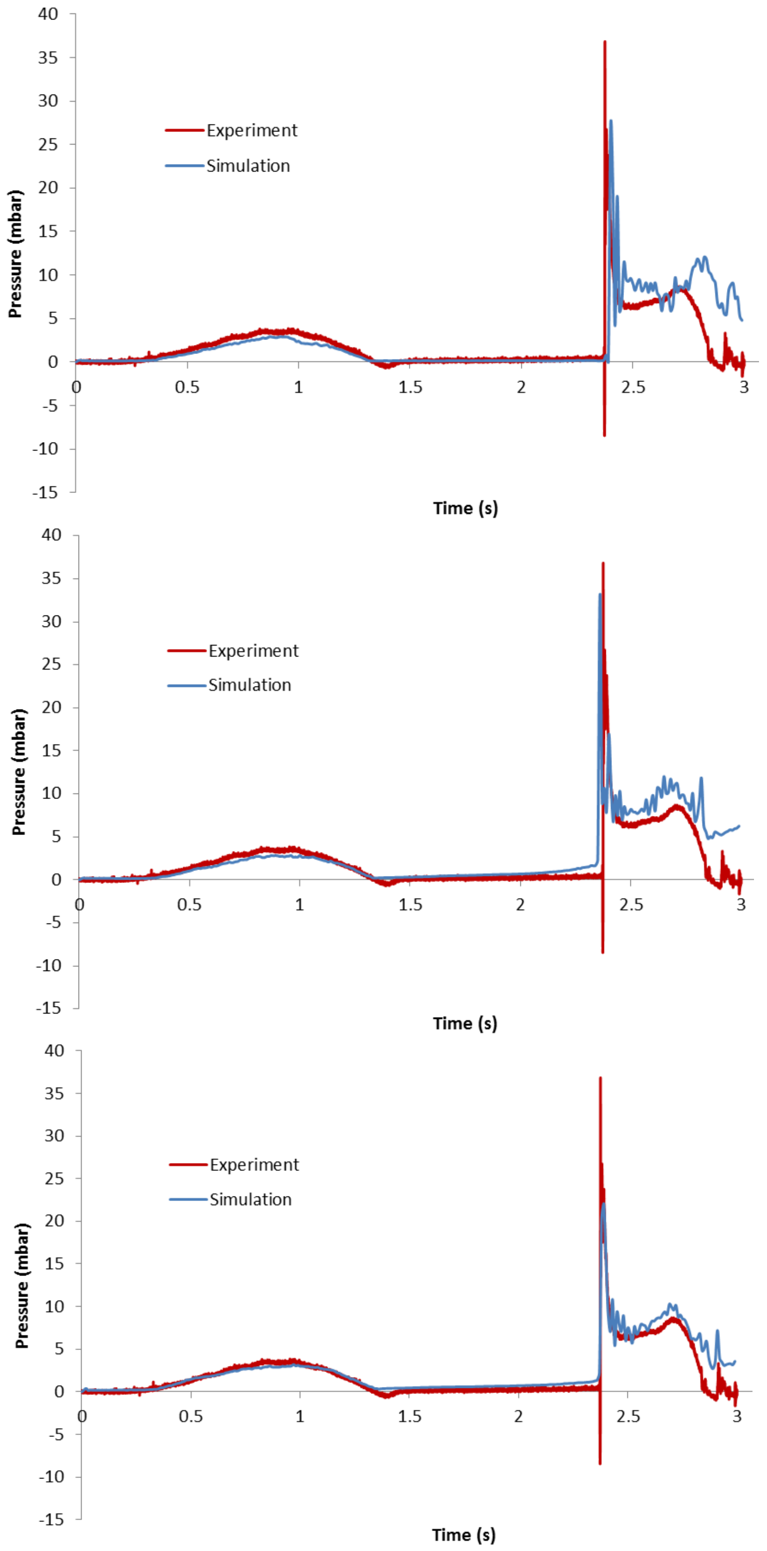
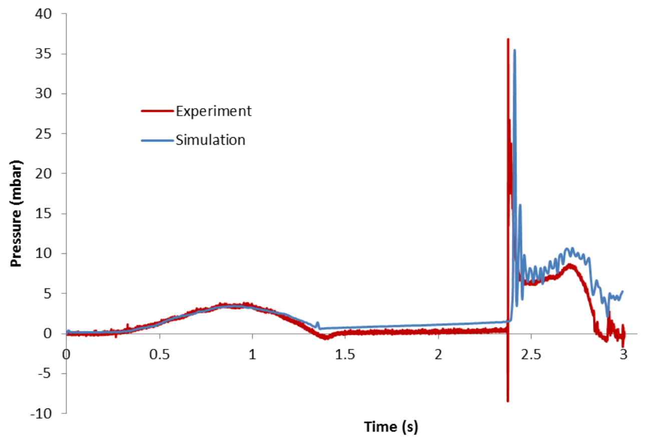


Figure 5.68: Comparison between experimental and simulation results for $c_{s,w}$=20m/s and for dx/h_0=0.0215. Investigation of the effect of the speed of sound ratio for an increased value for the water phase

a) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=5

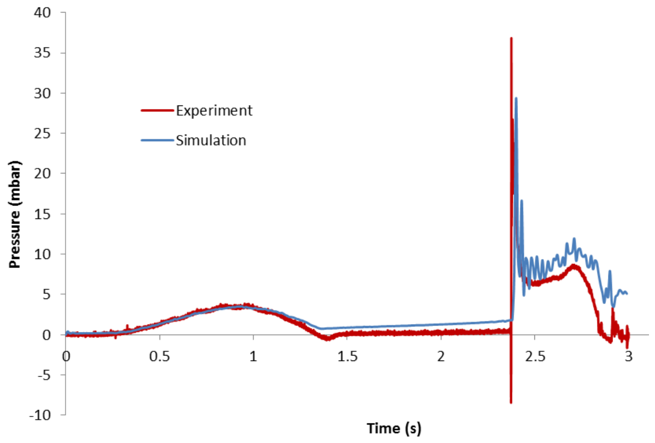
α=0.01



b) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=5

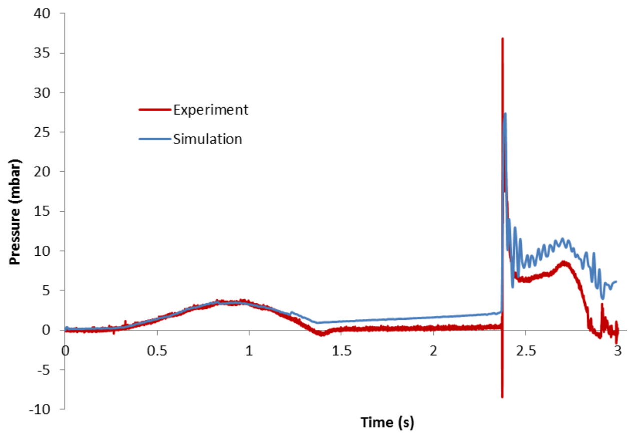
α=0.005



c) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=5

α=0.001



d) $c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=5

α=0.0005

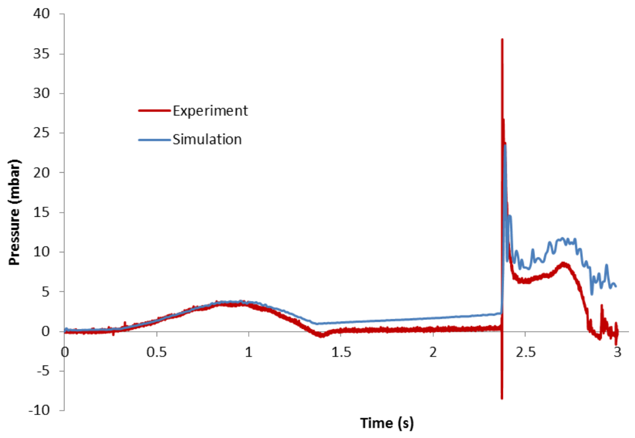
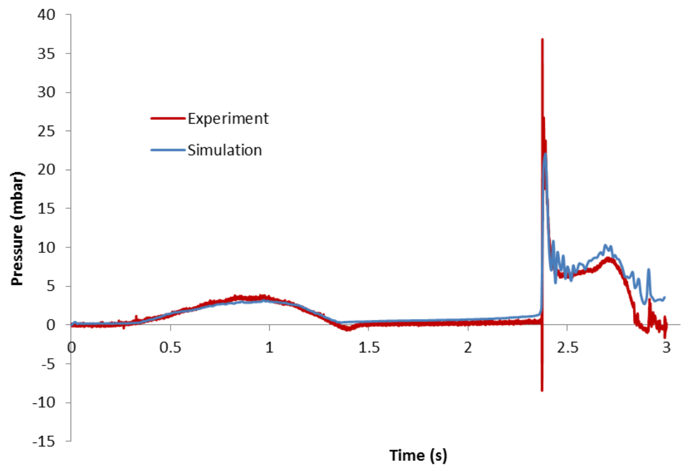
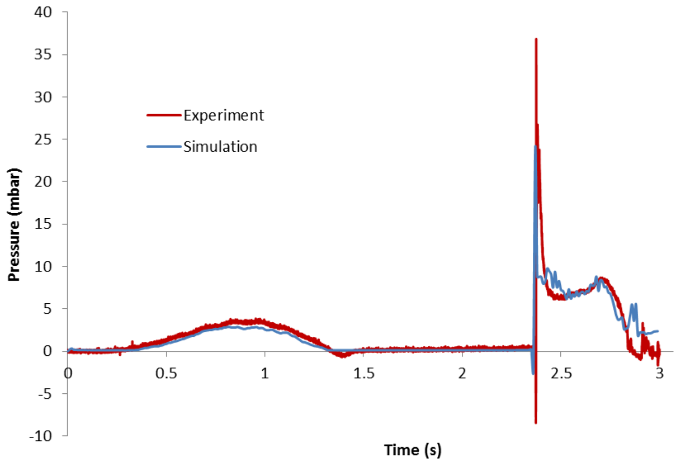


Figure 5.69: Comparison between experimental and simulation results for $c_{s,w}$=17m/s and for dx/h_0=0.0215. Investigation of the effect of the artificial viscosity

a) dx/h_0=0.0215



b) dx/h_0=0.0161



c) dx/h_0=0.0129

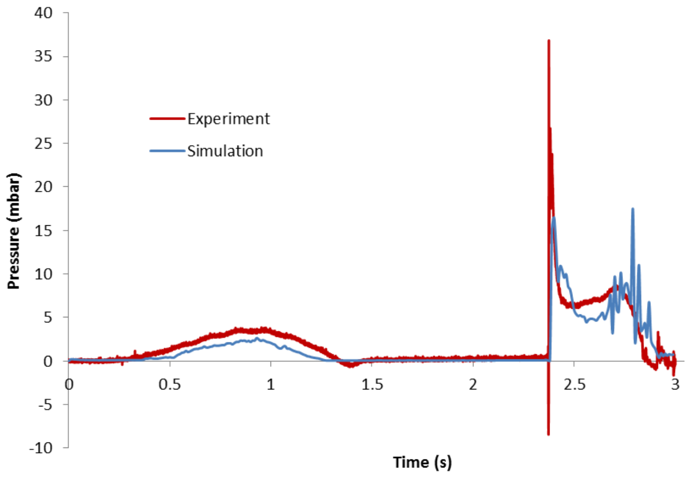


Figure 5.70: Comparison between experimental and simulation results for $c_{s,w}$=20m/s $c_{s,a}/c_{s,w}$=7.5 α=0.01. Investigation for different resolutions.

$c_{s,w}$=20m/s

$c_{s,a}/c_{s,w}$=5

α=0.0005

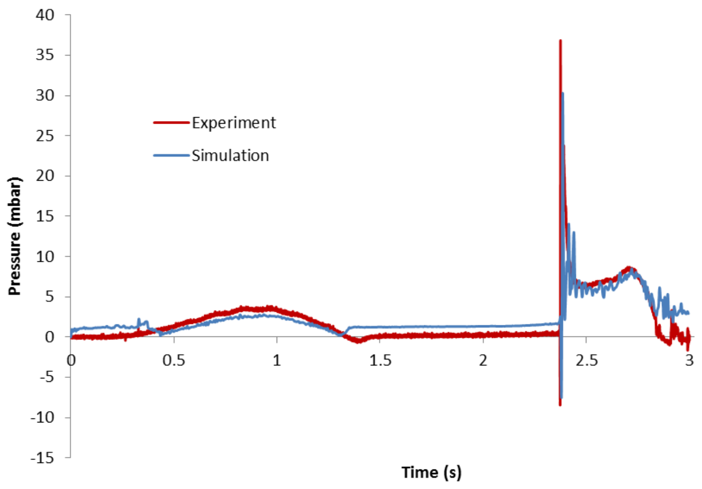


Figure 5.71: Comparison between experimental and simulation results for *dx/h*$_0$*=0.0161*

5.6. Concluding Remarks

The GPU-accelerated multi-phase SPH scheme has been tested against a range of demanding 2-D validation cases. A new particle shifting algorithm has been proposed that prevents the formation of unphysical voids in the air phase at high resolutions. For the simple formulation used for the GPU implementation, the scheme compares well with reference solutions for dry-bed dam break. For the wet bed dam break, similar to other published results, the gate movement needs to be delayed to replicate the flow. The agreement is close and further demonstrates the need for a multi-phase simulation for such a complex violent flow. A rolling tank case shows that the multi-phase SPH scheme can capture the high pressure peaks but similarly to other researchers, the results are sensitive to model parameters.

6. Validation: Three Dimensional Cases

6.1. Introduction

This chapter presents the validation of the multi-phase model extended to the third dimension with a complex 3-D flow. The extension was performed using the existing DualSPHysics functionality. The multi-phase model did not need to be adapted mathematically, but the optimisation detailed in Section 4.4 was created based on the performance of the code for both 2-D and 3-D simulations. The same code can then be used for both instances with the only difference being the extension of the surface term for the shifting algorithm. The code is validated comparing the results both to experimental data and existing single-phase simulations. Variables compared include the height of the water column at different points of the domain and the pressure.

6.2. Obstacle Impact

6.2.1. Case Description

This case was selected to investigate the ability of the code to model complex 3-D cases as well as provide experimental validation for the SPH scheme used. It is based on the experiments of Kleefsman *et al.* (2005) performed at the Maritime Research Institute of Netherlands and simulated a dam break case where the flowing water collides with a box. Due to this collision, the flow is inherently 3-D.

Waves hitting a stationary obstacle are a common issue in coastal or offshore engineering. Structures placed in areas near or in the sea must be designed in order to withstand any potential wave impact. Ships face the exact same issue and a material failure can result in the endangerment of the crew. One reported case occurred in November 1998 where the Schiehallion FPSO (Floating Production Storage and Offloading vessel) was damaged by a wave impact, forcing all personnel to abandon the ship (Gorf *et al.*, 2000).

Another issue occurring is the possibility of water entering the deck of the ship if the wave height is large. The volume of the water remaining on deck can be quite significant and follows the ship motion damaging the equipment and cargo stored in this area. The term green water is used to refer to this phenomenon and it can also impact offshore floating platforms (Kleefsman *et al.*, 2005). Green water loads are particularly important for designing the local support structure (Pauling, 2008).

The experiment is a simple model of a green water flow with the obstacle simulating a container on the deck of a ship. It is a similar case to the dry dam break problem described in Section 4.8 and 5.2. It has been simulated by several researchers using a single-phase SPH model since 2010 (Lee *et al.*, 2010), including simulations using GPU acceleration by Crespo *et al.* (2011a) and Rooney *et al.* (2011). It has not yet been simulated using a multi-phase approach.

The experimental setup uses a lock gate to hold the volume of water motionless at the beginning of the case. The gate is raised using a weight system. The raise is almost instantaneous with the water flow not being affected by the gate, so its motion will not be included in the simulation. The water will then flow towards the obstacle which is firmly set in place.

The main point of interest is the water movement and especially its interaction with the box. During the experiment measurements were performed at the height of the water flow as well as the pressure and forces on the box. Four height probes were used with the first being in the reservoir, while the box was covered by eight pressure sensors. The position of the height probes and the pressure sensors are shown in Figure 6.1.

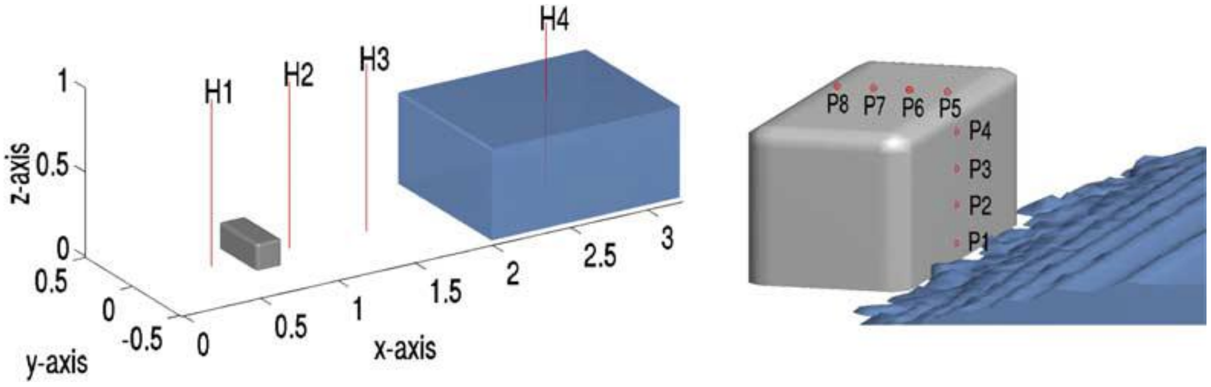


Figure 6.1: Measurement positions for water heights and pressures in the experiment (Kleefsman *et al.*, 2005)

When simulating the experiment using DualSPHysics with a single-phase model, Crespo *et al.* (2011a) compared their results with the experimental ones for the first three height probes as well as two of the points measuring the pressure on the side of the box. The measure points are shown in Figure 6.2 along with the detailed geometry of the tank:

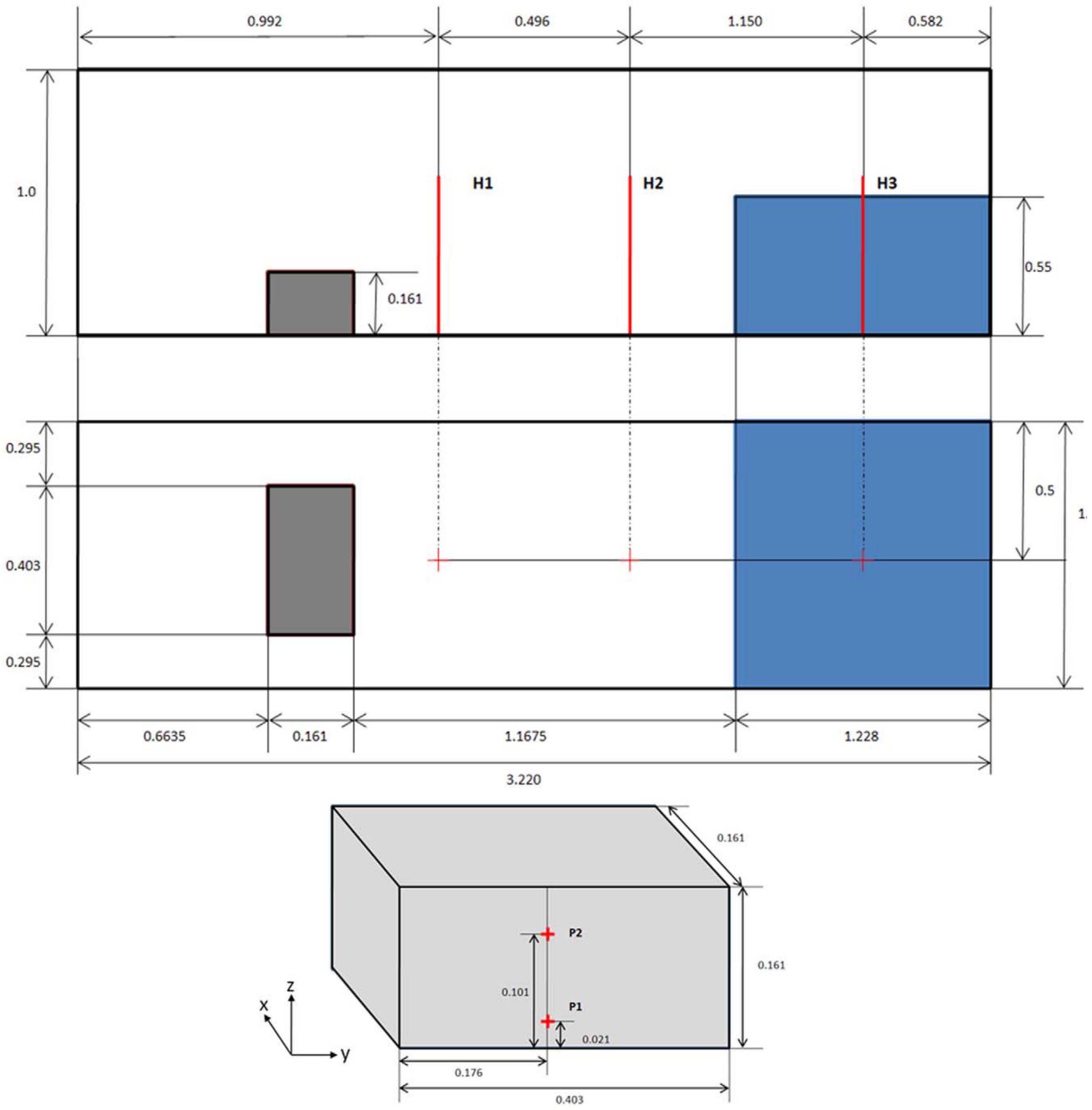


Figure 6.2: Defining sketch of the domain and the experimental measuring positions (Crespo *et al.*, 2011a)

In general, a 3-D case is a much more demanding simulation especially for an air-water flow. The air particles vastly outnumber the water phase, so if the collision with the box is to be modelled accurately, a significantly larger particle number is needed. The tank will be modelled using the exact dimensions of the experiment; the rapid advancement of the water flow towards the obstacle and the small size of the tank mean that the domain cannot be shrank while retaining identical Reynolds and Weber flow numbers. The initial profile of the water flow can be seen in Figure 6.3.

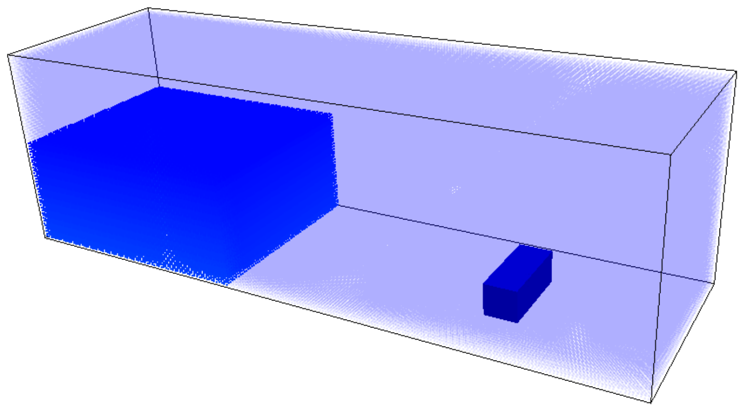


Figure 6.3: Initial domain of benchmark case 2

6.2.2. Validation

Figure 6.4 to Figure 6.6 present the evolution of the flow for a simulation ran with one million particles which leads to particle spacing dx/h_0=0.0273, where h_0 is the initial height of the water in the tank has been executed and dx the initial particle distance. The shifting algorithm is used for both phases with the water being treated by the free-surface correction. The artificial viscosity coefficient is 0.01 and the speed of sound ratio is 10.

After the start of the simulation the water column collapses under the effect of the gravity and moves towards the negative *x*-direction of the domain, where, about 0.5s after the beginning of the simulation, the flow collides with the obstacle reducing the velocity of the toe particles in the middle of the flow to 0. The remaining particles continue their movement until they reach the opposite wall. Figure 6.4a shows the beginning of the collapse of the water column while Figure 6.4b shows an instance of the simulation shortly after the water flow reaches the obstacle.

When the particles at the sides of the box that did not collide with the obstacle reach the opposite wall they move upwards as a result of the forces exerted by the still moving part of the flow. At the same time the central part overflows the obstacle creating a plunging wave as seen in Figure 6.5a. The plunging wave breaks on the back wall covering the area behind the obstacle while the rest of the flow continues its forward motion.

The forward motion is eventually halted by the back wall with the majority of the water particles being gathered in this area, creating a second shorter column. This column also collapses creating a wave travelling in the opposite direction, a reflected wave. The wave evolution movement is shown in Figure 6.6 for three different instances. A slight bump of the water flow over the obstacle is visible in all three screenshots.

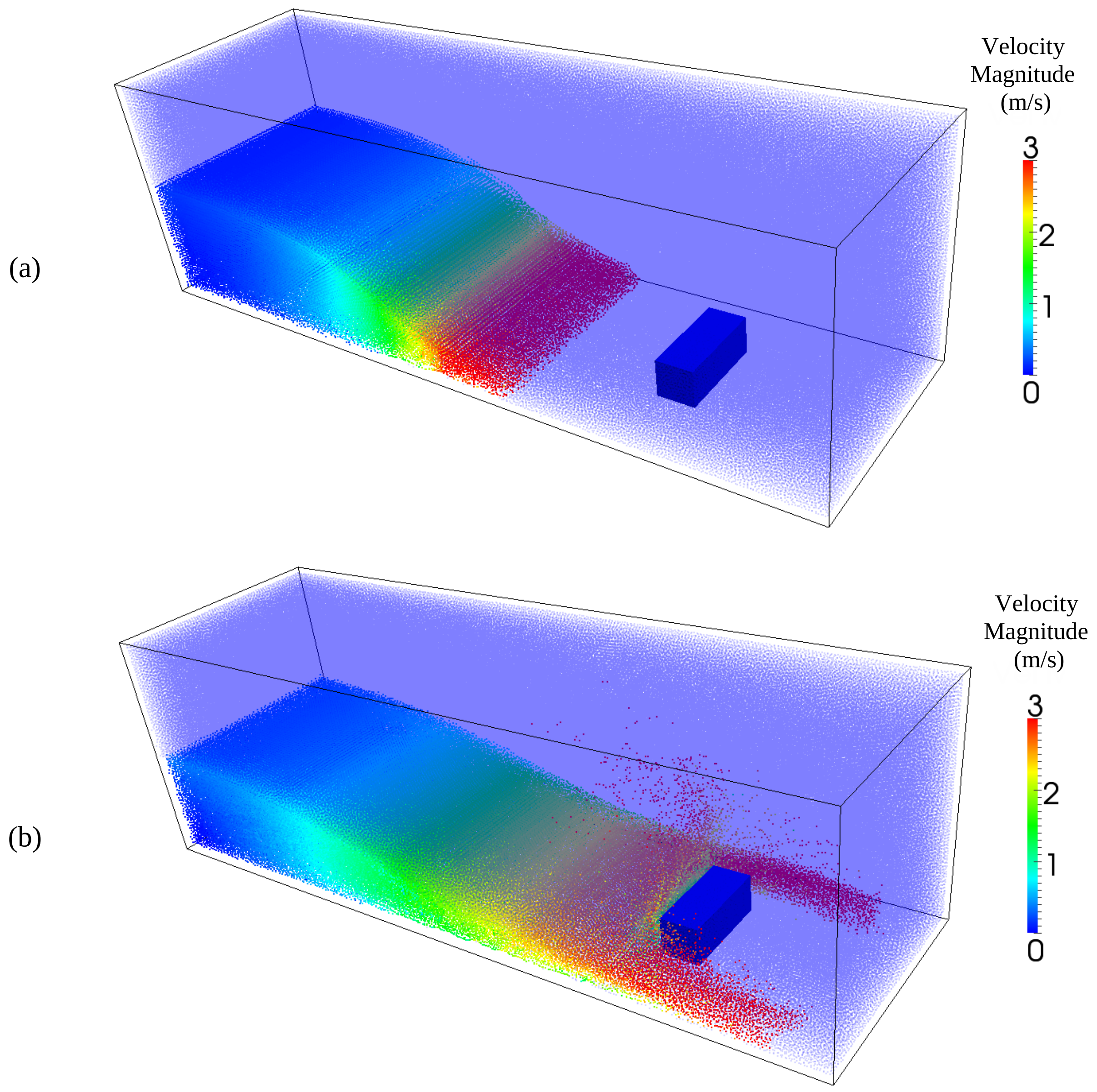
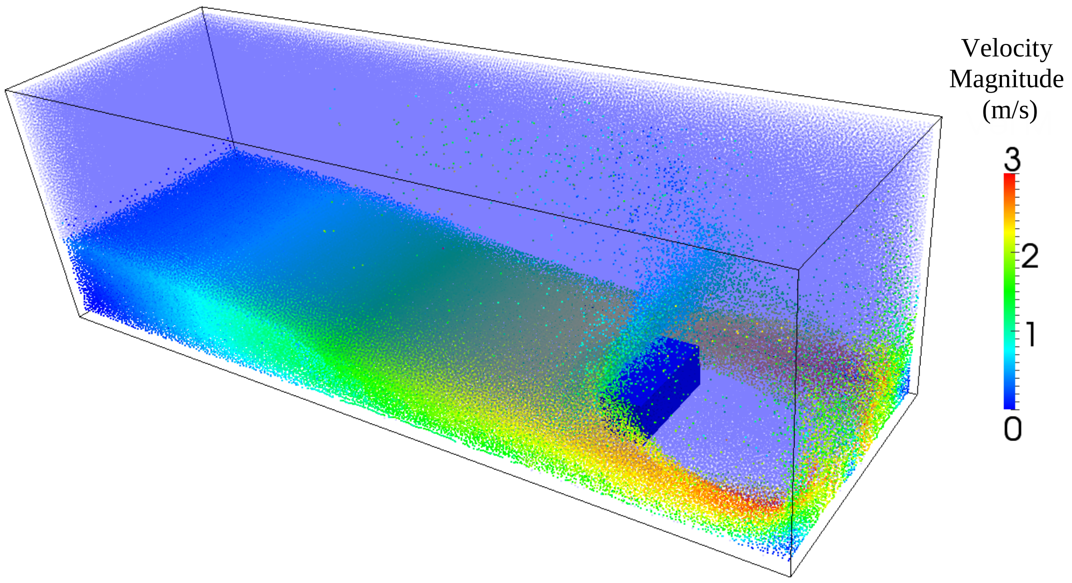


Figure 6.4: Velocity magnitude contours of a 3-D obstacle impact flow for one million particles with spacing *dx/d_0*=0.0273 at a) 0.3s and b)0.6s

(a)



(b)

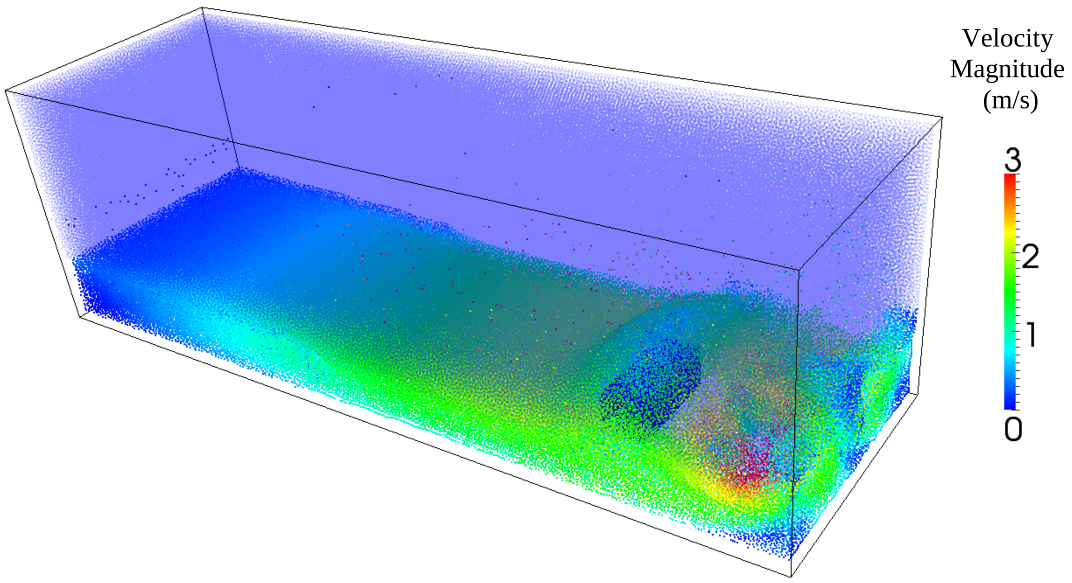


Figure 6.5: Velocity magnitude contours of a 3D obstacle impact flow for one million particles with spacing *dx/d_0*=0.0273 at a) 0.9s and b)1.2s

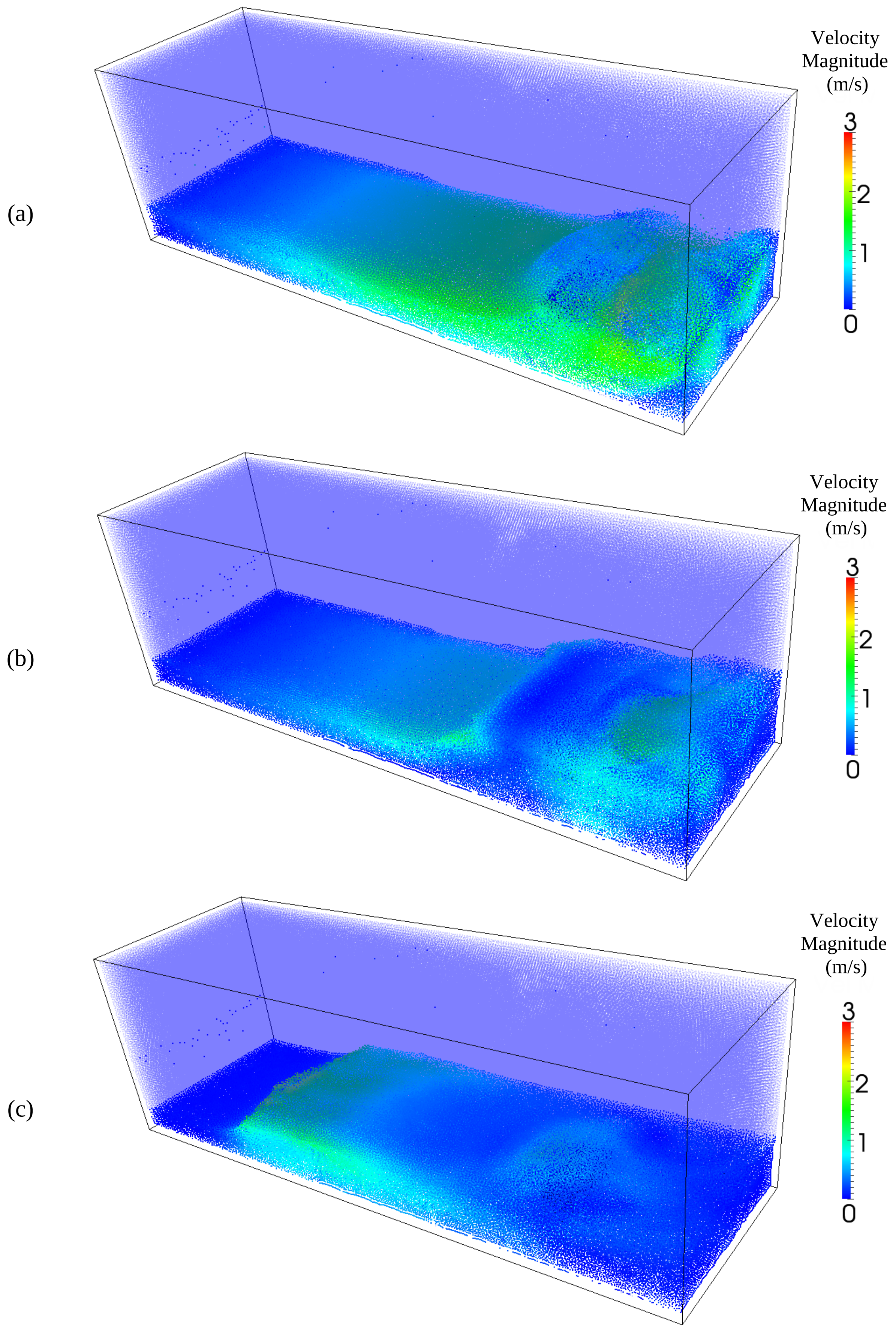


Figure 6.6: Velocity magnitude contours of a 3D obstacle impact flow for one million particles with spacing dx/d_0=0.0273 at a) 1.5s b)2s and c) 3s

Kleefsman *et al.* (2005) have provided an animation of the experiment from which instances at different points can be extracted. The difficulty in distinguishing the water from its background and the creation of foam make a direct comparison between the simulation and the experiment very difficult. Nevertheless, Figure 6.7 shows a comparison between the multi-phase SPH simulation and two snapshots from the experiment.

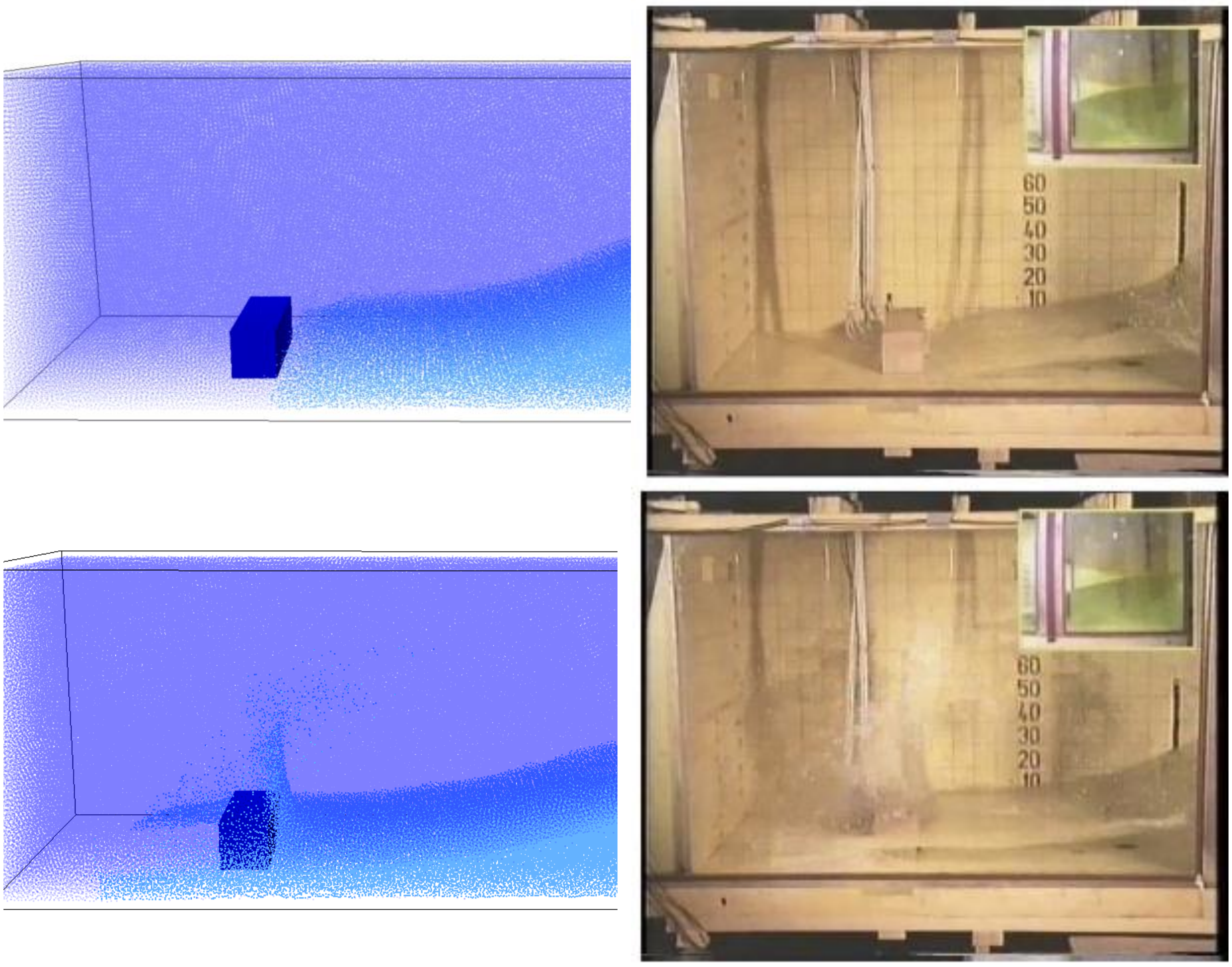


Figure 6.7: Comparison of the multi-phase simulation and the experiment at 0.4s and 0.56s

Apart from the animation, detailed data regarding the pressure and height measurements of the experiment are available. Figure 6.8 to Figure 6.11 show the results for the height of the water. The measurement positions follow the height probes by Kleefsman *et al.* (2005). The case has been executed with two multi-phase simulations; one using the shifting algorithm and one without it. A single-phase simulation using version 2.0 of DualSPHysics has also been conducted.

The shifting algorithm has been applied to both phases; however, the surface term given by Equation (5.23) and therefore taking into account the third dimension is only applied to the water phase. The speeds of sound used for the phases are calculated using the same approach as the dry dam break case, being calculated as $12\sqrt{gh_0}$. A ratio of 10 was selected between the air and the water speed of sound with the water value set at 30m/s. The artificial viscosity coefficient is calculated using the approach described by Monaghan and Kajtar (2009b). Specifically, the artificial viscosity parameter can be linked to the kinematic viscosity via the following Equation:

$$v = \frac{1}{8} a h \bar{c}_{s,ij} , \tag{6.1}$$

where v is the kinematic viscosity and $\bar{c}_{s,ij}$ is the average speed of sound between particles i and j. The kinematic viscosity and Equation (6.1) can then be linked to the Reynolds number, which is computed using the initial height h_0 of the water column as characteristic length and $\sqrt{gh_0}$ as its characteristic velocity. The smoothing length used in this case is the same as Crespo *et al.* (2011a) used for their single-phase 100,000 particle simulation equal to 3.075×10^{-2}m. That corresponds to a particle distance dx/h_0=0.0364.

Figure 6.8 shows the results for height probe H4. Its position is in the middle of the initial water column, so the initial value is the starting height of the water. The height is gradually reduced until close to the end of the simulation, where the reflected wave appears. All the simulations follow the experimental results closely, but they do not predict the reflected wave correctly; its arrival is delayed. The closest results to the experiment are given by the multi-phase simulation without the shifting algorithm, while the other simulations give almost identical results.

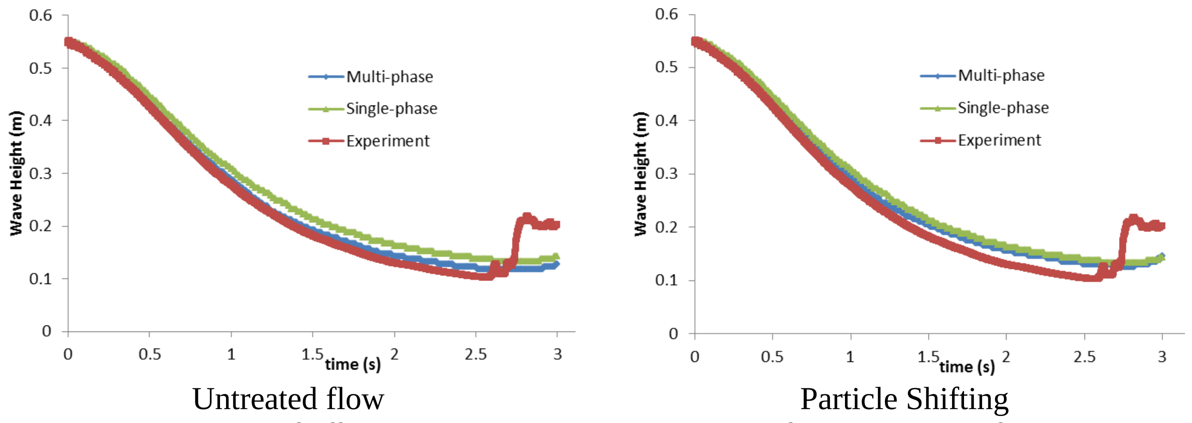
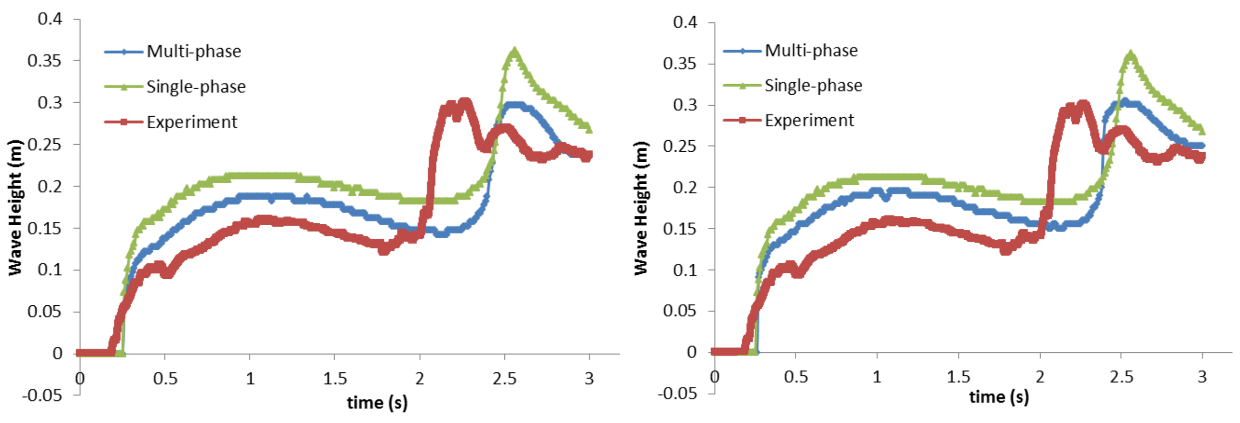


Figure 6.8: Comparison of different simulations and experimental results for height probe H4 for dx/h_0=0.0364

The second probe, H3, is located at the space between the initial water column and the obstacle. The height of the flow at this point remains constant until the reflected wave appears. From Figure 6.9 we can see that there is a discrepancy between the simulation results and the experimental results even before the arrival of the returning wave. This is a boundary issue that has already been mentioned in the dry dam break case: the fluid particles are pushed upwards by the boundary particles as they begin their movement. This occurs due to the small number of water particles present at this point which leads to an incomplete kernel. The height difference between the single-phase and the experiment is ~2*h*, while the difference for the multi-phase models is about ~h.



Untreated flow

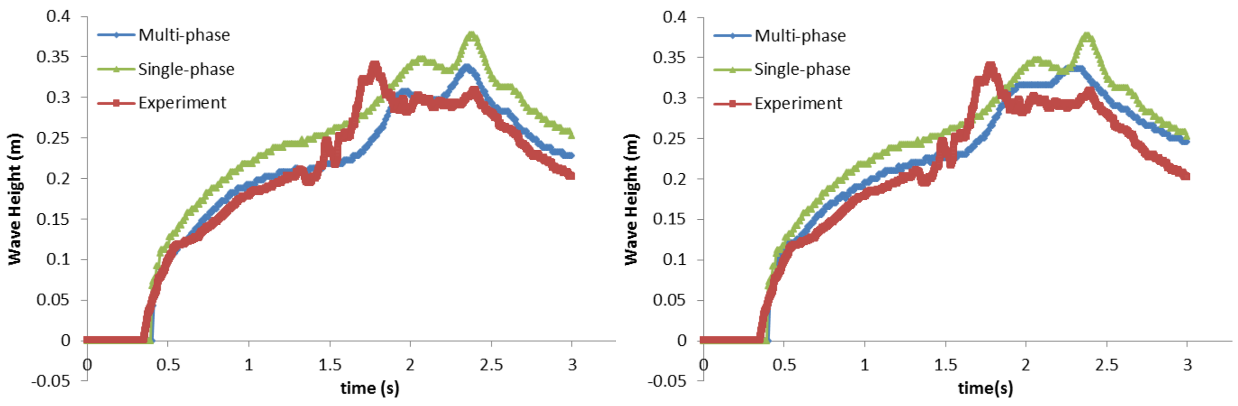
Particle Shifting

Figure 6.9: Comparison of different simulations and experimental results for height probe H3 for *dx/h*$_0$=0.0364

It is less pronounced for the multi-phase simulations due to the opposite force exerted by the air particles which reduce the upwards movement caused by the boundaries and which also complete the kernel for the water phase leading to a smaller error. Apart from this issue the discrepancy for the reflected wave can also be observed by this probe; there is a delay in the height increase expected. The expected height is however, better captured by the multi-phase simulations.

Height probe H2 is placed right before the obstacle, so the water flow reaches this point just before the impact. This probe shows the gradual build-up of water at the end of the vessel, which is then followed by the reflected wave. Results are shown in Figure 6.10 and the multi-phase simulations follow the result closely, while the single-phase model shows the same issues as in probe height H3, an increase in height due to the effect of the boundaries.

The height increase by the reflected wave is not predicted by either of the simulations which only show a constant increase. The subsequent plateau is followed by the simulations but the small height increase at around 2.5s is overestimated. The closest results are obtained again by the multi-phase simulation without the shifting algorithm, with the other two simulations constantly overestimating the height.

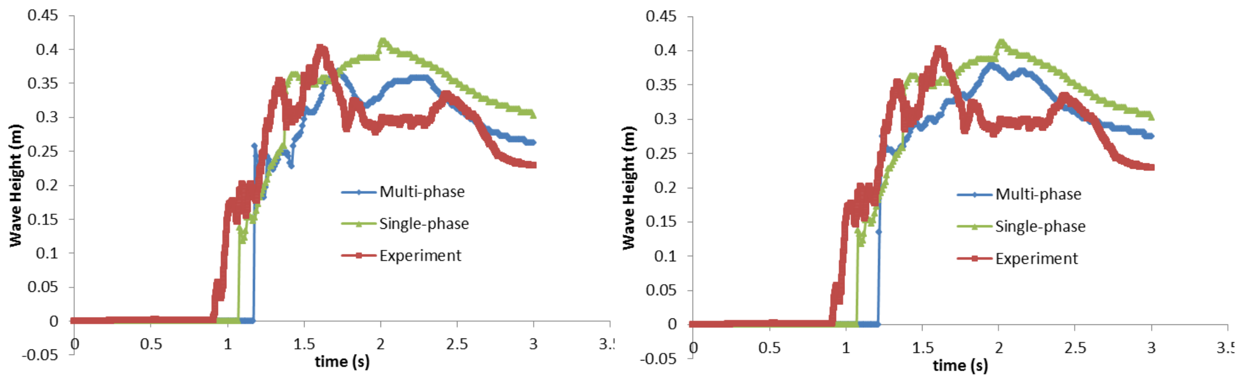


Untreated flow | Particle Shifting

Figure 6.10: Comparison of different simulations and experimental results for height probe H2 for dx/h_0=0.0364

For the last height probe, H1, which is situated at the back of the obstacle, agreement with the experimental results is less accurate, as shown in Figure 6.11. A reason for that is the phase delay for the reflected wave observed in the previous height probes. This is the area where the wave originates, so the delay is more prevalent here affecting every aspect of the computation in the area. The results here are closest to the experimental ones for the single-phase model although the height at the end of the computation is overestimated.

A secondary reason is the recirculation of the particles towards the centre of the domain after the particles hit the opposite wall. This behaviour, which causes the gradual height increase around 1s, is delayed for the multi-phase simulation. The delay is partially due to the boundaries as it also affects the single-phase simulation but is amplified by the resistance the air particles pose to the water movement.



Untreated flow   Particle Shifting

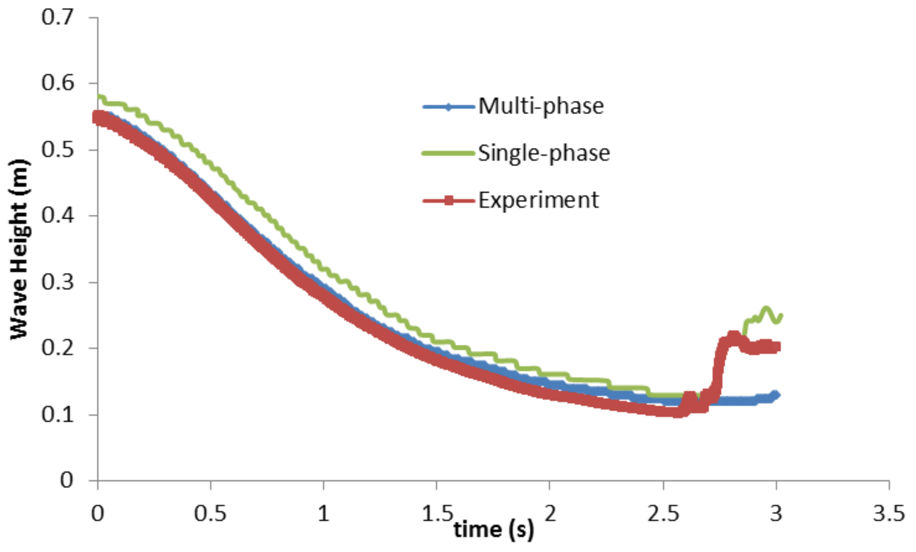
Figure 6.11: Comparison of different simulations and experimental results for height probe H1 for *dx/h₀*=0.0364

Between the multi-phase simulations with and without the shifting algorithm, a difference in the predicted height can be seen. In general, without the shifting algorithm the simulation predicts a lower height, closer to the experimental results, while the results of the other model are closer to the single-phase simulation. The shifting algorithm prevents increases in particle concentration and maintains a more even particle spacing. When subjected to the boundary forces, this moves the entire flow slightly upwards as seen in Figure 6.9 as due to the effects of the free surface term, the effect of the force exerted by the air particles is diminished. This height increase is then transferred throughout the phase so that particles can maintain their spacing and leads to the interface being shifted slightly upwards.

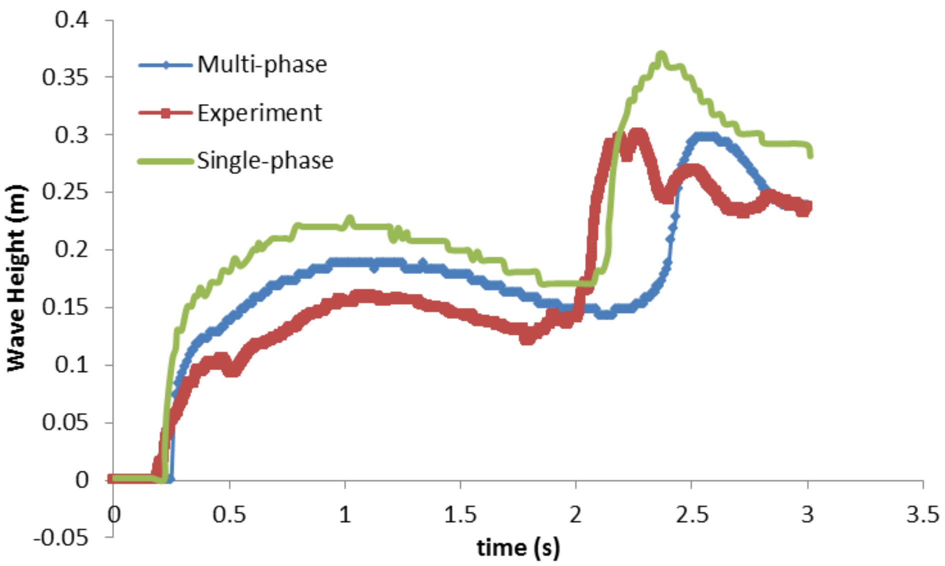
At this resolution, no voids in the domain were seen for the untreated flow. A smaller difference between the two is the pressure on the water spray after the impact; lone water particles in the untreated flow are characterised by oscillatory pressure with very high amplitude. This is not the case if the shifting algorithm is used, with lower pressure values.

The height results presented here will also be compared with the results of Crespo *et al.* (2011a). They simulated this case using three different resolutions, the middle of which corresponds to multi-phase computations performed in this study and discussed earlier. Compared to this study, Crespo *et al.* (2011a) use the Verlet time-stepping algorithm instead of the predictor-corrector model and the XSPH velocity correction, which is not used in this study. The results are presented in Figure 6.12 for height probes H4, H3 and H2 (they are referred in their paper as H3, H2 and H1 respectively). Results for height probe H1 are not shown. The results will be compared with the multi-phase simulation without the shifting algorithm.

H4



H3



H2

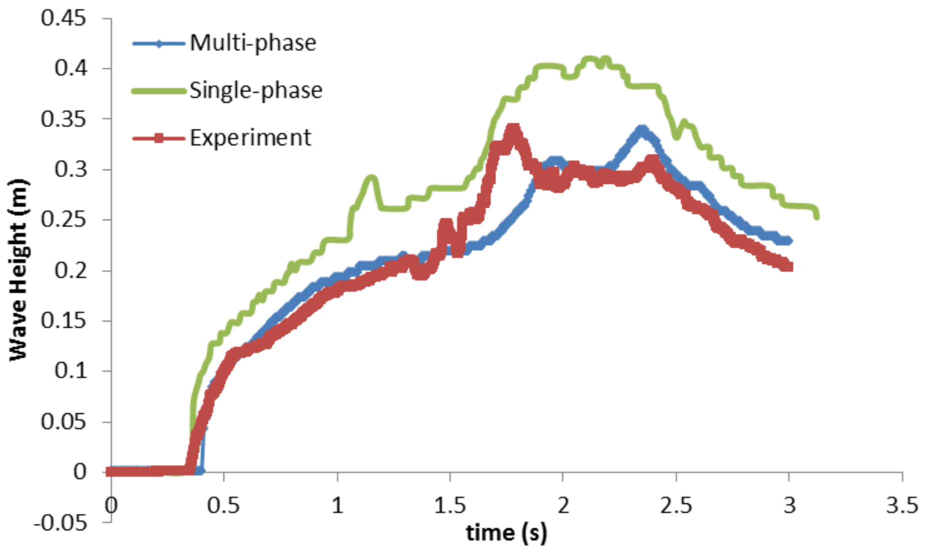


Figure 6.12: Comparison of a multi-phase simulation and the results of Crespo *et al.* (2011a) with the experimental results for dx/h_0=0.0364

As can be seen in Figure 6.12, the results by Crespo *et al.* (2011a) show a better agreement regarding the timing of the reflected wave. However, the height of the water flow is always overestimated, even compared to the single-phase results from Figure 6.8 to Figure 6.11, with the multi-phase model having a better agreement. The results of Crespo *et al.* (2011a) were obtained using an older version of DualSPHysics, so these improvements show the evolution of the code in the meantime.

The obstacle impact case has also been executed with a higher resolution, using smoothing length 2.25×10^{-2}m or resulting in particle distance dx/h_0=0.0272 which for these dimensions translates to dx=0.015m. The results are presented in for a single and a multi-phase case with this resolution which corresponds to about 250,000 water particles, 750,000 air particles and over 1 million particles in total.

Figure 6.13 show the height for probes H4, H3 and H2 and there is no significant difference for the untreated flow with the lower resolution. The shifting algorithm approach is slightly improved for probe H3 and H2 producing similar results to the untreated flow although it is still not as close to the experimental results in the latter parts of the flow. For probe H1 the delay is still noticeable, but the behaviour of the flow after is significantly better than the one demonstrated in Figure 6.11 for the lower resolution and closer to the experimental results.

The obstacle impact case was also simulated halving the particle distance used in the simulations presented in Figure 6.12 resulting in particle distance dx/h_0=0.0182 or for these dimensions dx=0.01m. Since this is a 3-D simulation the number of particles has been increased by 8 times, reaching about 4 million particles. The results will be compared to the multi-phase simulation presented in Figure 6.12 as well as the experimental results. The simulation has been executed for 2s and the results are presented in Figure 6.14.

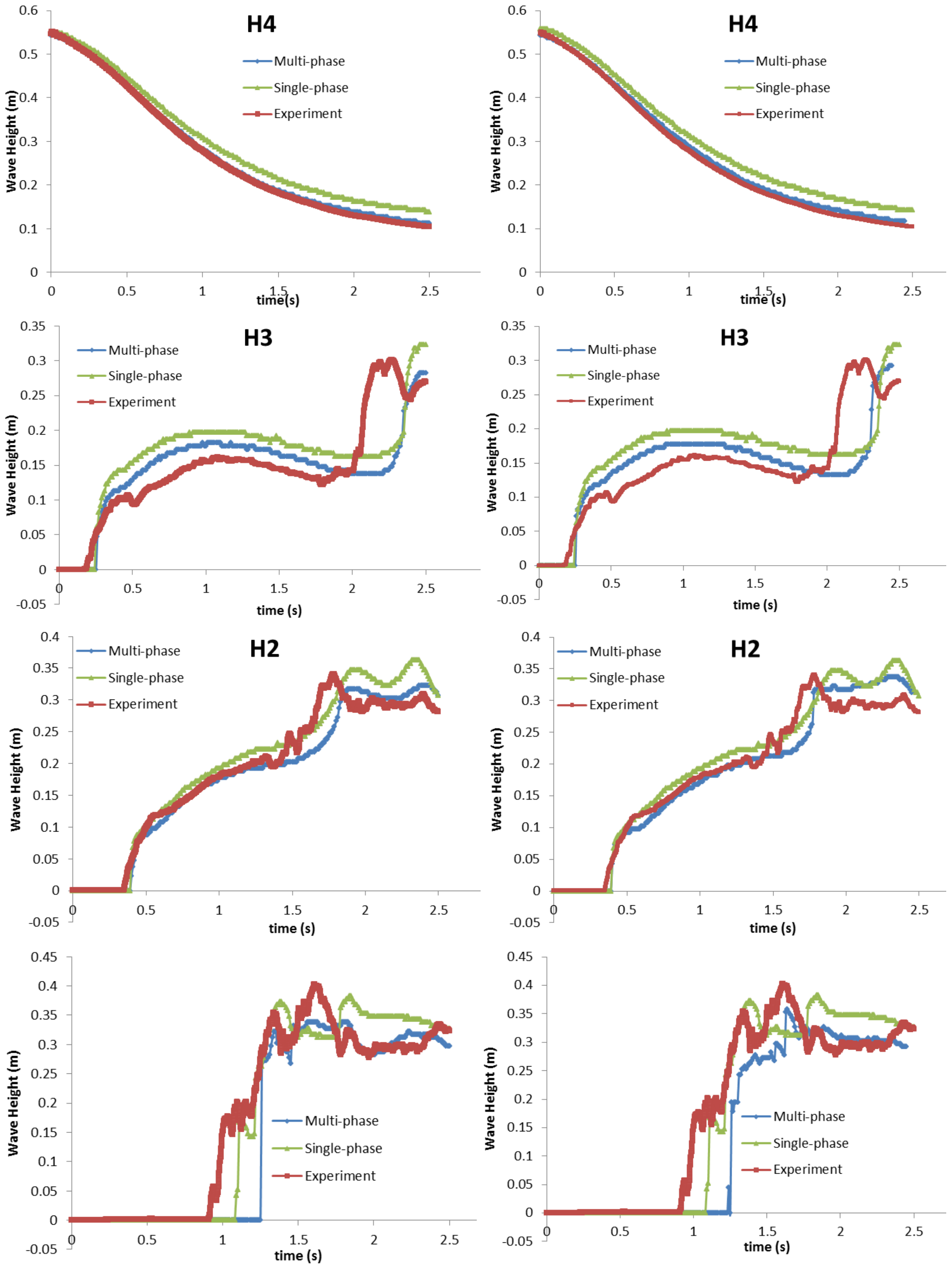
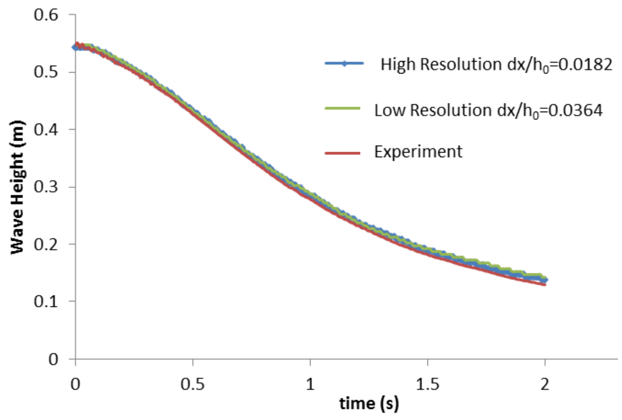
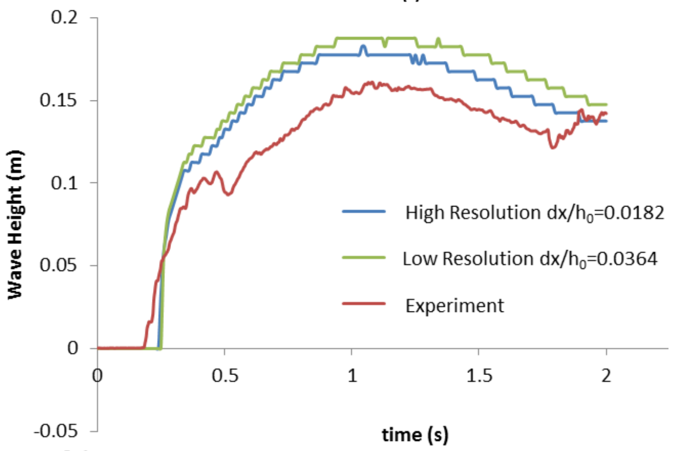


Figure 6.13: Comparison of multi-phase simulations with (right) and without (left) the shifting algorithm and a single-phase simulation with the experimental results for dx/h_0=0.0273

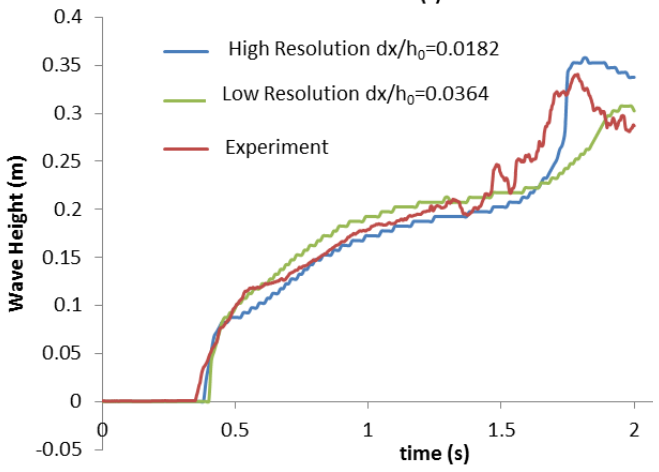
H1



H2



H3



H4

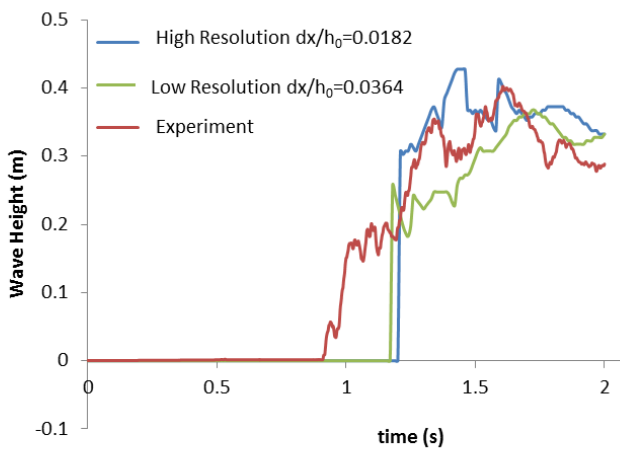


Figure 6.14: Comparison of height results for 5×10^5 and 4×10^6 particles with experimental results

Results from probe H1 show no difference between the two simulations, with them having a very good agreement with the experimental results. Differences start occurring at probe H2 with the effect of the boundary particles being lessened. The reason for that is the smaller smoothing length and particle distance in the larger simulation. Neither of the simulations however, predicts the returning wave at the correct moment, showing a delay witnessed in all the numerical simulations performed in this study.

Height probe H3 shows the greatest improvement between the two simulations. The rapid height increase observed after 1.5s due to the reflected wave is simulated by the high resolution computation, albeit with a slight delay. The height at the end of the simulation is however over predicted. Probe H4 however, shows large differences between the simulations and the experiment. A large part of the difference is due to using the mass for the measurement of the height; instead of the free surface the spray created by the impact is being tracked. This can be determined by the instant height increase at 1s and the constant decrease thereafter.

Apart from the heights in the domain, experimental data for the pressure in the designated probes is available. The experimental results will be compared with the numerical ones obtained at the previously detailed resolutions. The results are presented in Figure 6.15 for the four probes in the face of the obstacle. Two sets of results are presented, with and without particle shifting. The multi-phase simulation considered are the same as Figure 6.12 with particle distance dx/h_0=0.0364 leading to a particle number close to 500,000.

The results presented show that the pressure peak at the beginning is underestimated while being overestimated for the latter half of the simulation. A delay between the initial pressure peak in the simulation compared to the experiment can also be seen. The simulation pressure follows however the experimental results with the exception of the fourth probe, where the pressure peak is not recorded.

An interesting aspect of the simulation is the amplitude of the oscillations, which is smaller when using the shifting algorithm. This is due to the reorganising of the particle position in order to maintain a zero concentration gradient, leading to a more uniform domain. This is presumably the reason for the lower pressure peaks shown for the shifting simulation.

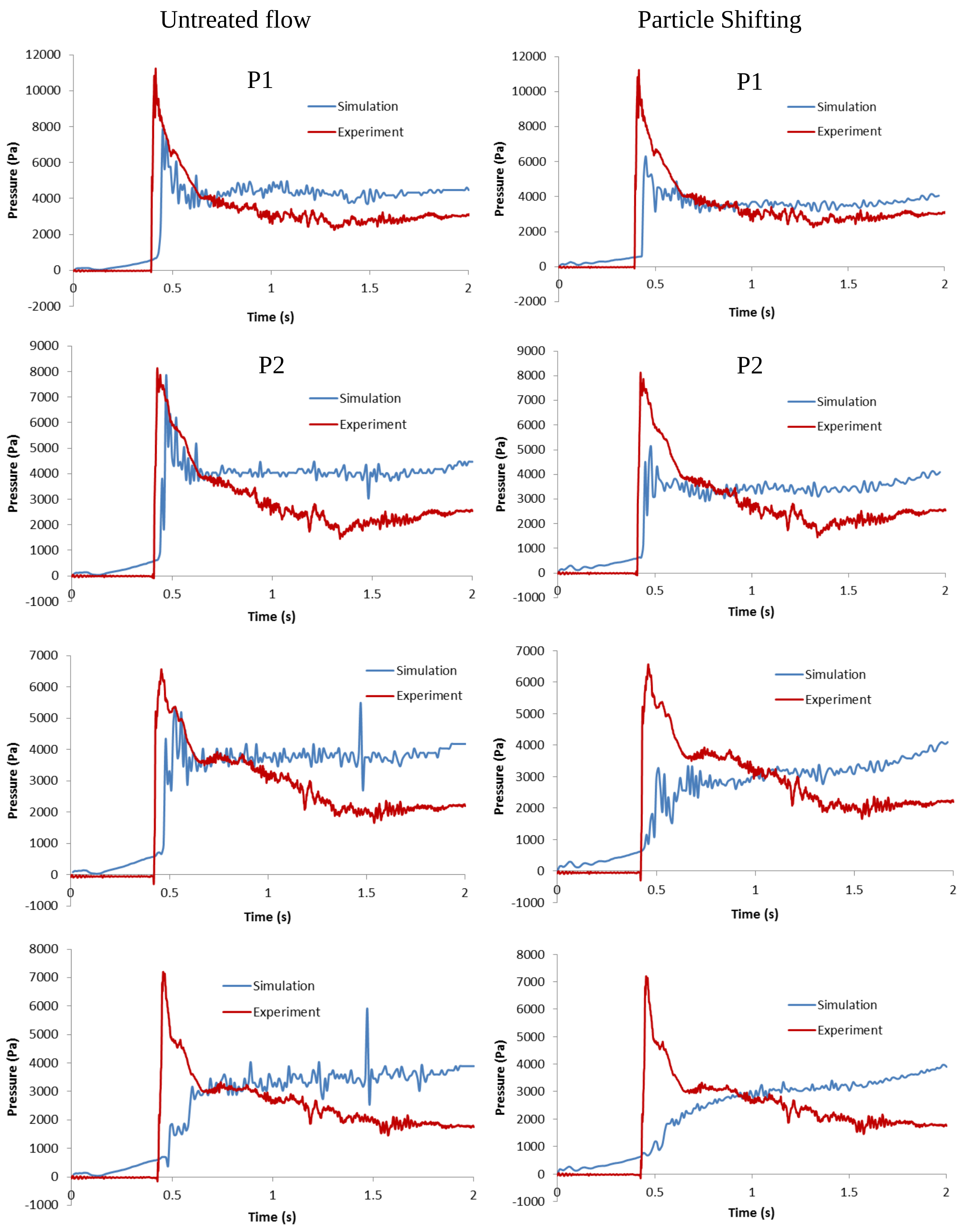


Figure 6.15: Comparison between experimental and simulation results for the 4 pressure probes in the side of the obstacle for two simulations with dx/h_0=0.0364

To investigate the effect of the speed of sound for the pressure, two additional simulations were run. In both simulations, the speed of sound for the water phase was lowered by about 30% and set to 20m/s. The speed of sound for the air either remained the same value as before, resulting in a ratio of 15, or also lowered by 30% resulting in a ratio of 10. The results are presented in Figure 6.16.

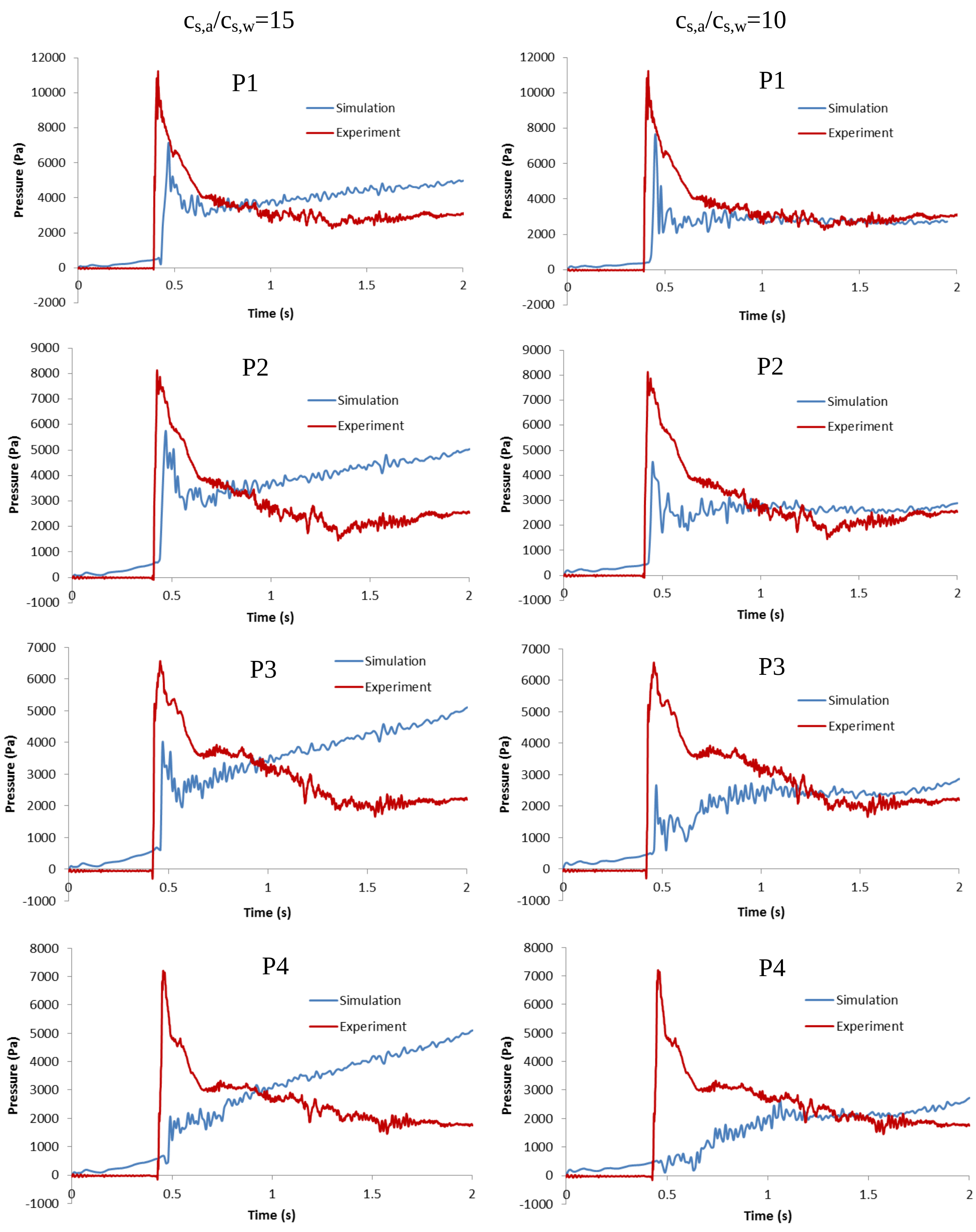


Figure 6.16: Comparison between experimental and multi-phase simulation with the shifting algorithm results for the 4 pressure probes in the side of the obstacle for two different speed of sound ratios

The results of Figure 6.16 show that the speed of sound for the air phase has a much greater effect on the pressure peak than the speed of sound of the water or the ratio between them. When the value remains unchanged, the pressure peak values are much closer to the results displayed in Figure 6.15, despite the decrease in the water value. In comparison, when the air value is decreased the pressure peak results are also decreased.

The latter half of the simulation also depends greatly in the speed of sound for the air phase with the pressure settling for a much lower value. The speed of sound ratio however, has a greater effect on that part. Compared to Figure 6.15, the final pressure value as well as the rate of change is greater with the higher ratio. The pressure values have not stabilised either; there is a significant increase even at the end of the computation.

Neither of the simulations however, is able to predict the peak pressure closely, as well as model it for probe P4. For that reason the pressures have also been computed with two higher resolutions, one doubling the number of particles with dx/h_0=0.0273 and the other halving the particle distance with dx/h_0=0.0182. Simulations using the first resolution have been executed both with and without applying the shifting algorithm, while the higher resolution is only referenced without it. The simulations have been investigated when comparing the particle heights in Figure 6.13 and Figure 6.14 respectively.

For dx/h_0=0.0273 in Figure 6.17, the untreated flow shows greater pressure oscillations, especially when looking at the results of probe P3 and P4. Compared to the low resolution results however, shown in Figure 6.15 the oscillation amplitude is much smaller. The results for the pressure peaks for the two simulations are also closer to the experimental ones, especially for the shifting simulation, which now produces results similar to the untreated flow.

For the higher resolution of dx/h_0=0.0182 in Figure 6.13 and Figure 6.18, the results for the pressure peaks and generally the first half of the simulation are very close to the experimental results showing a definite improvement over the smaller resolutions. However, the pressure decline and its residual value are not captured correctly with the simulation overestimating the experimental values. As observed in Figure 6.16 the speed of sound and the ratio have a significant impact on the final pressure values and it is possible for these discrepancies to be corrected if the numerical parameters are better defined.

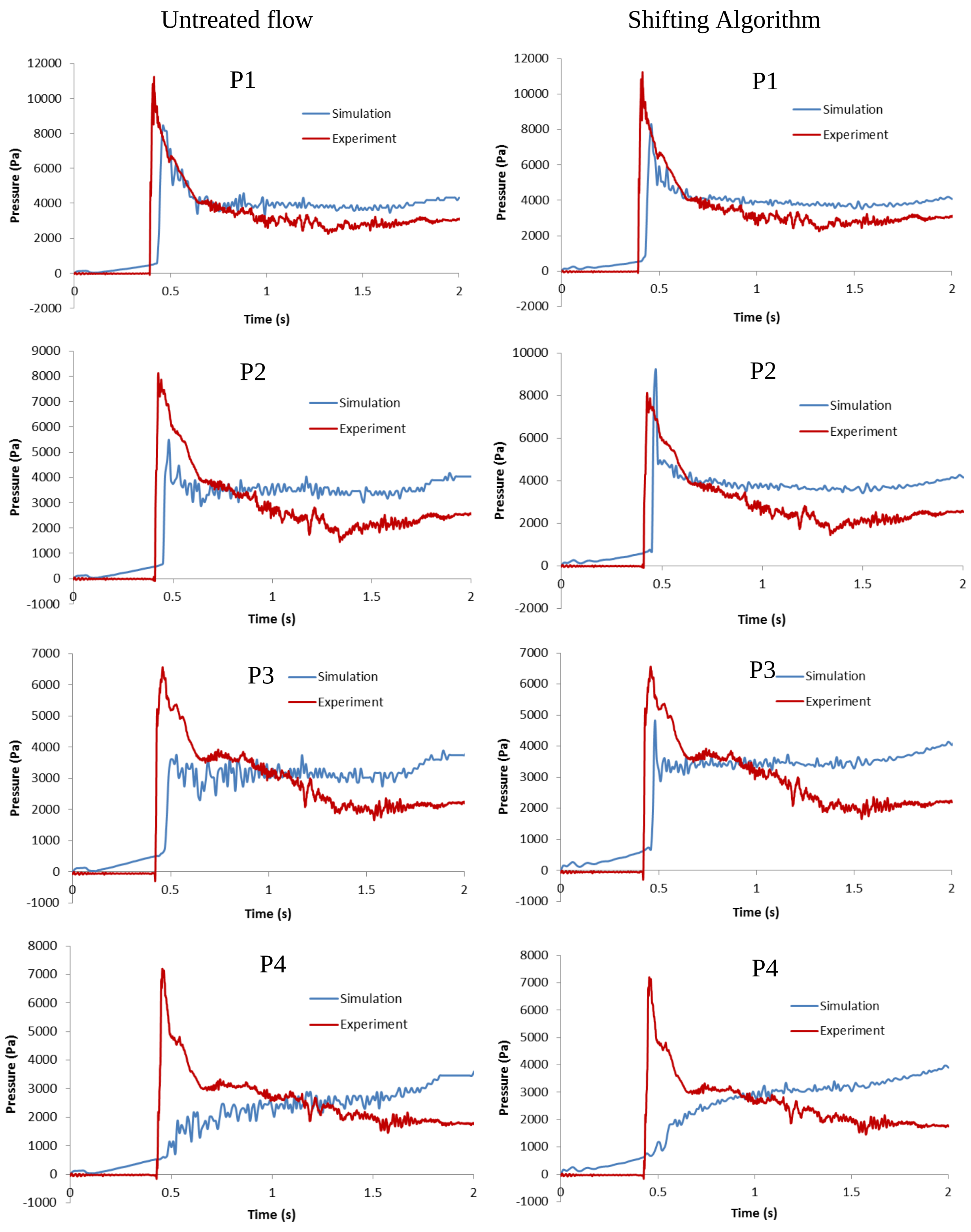
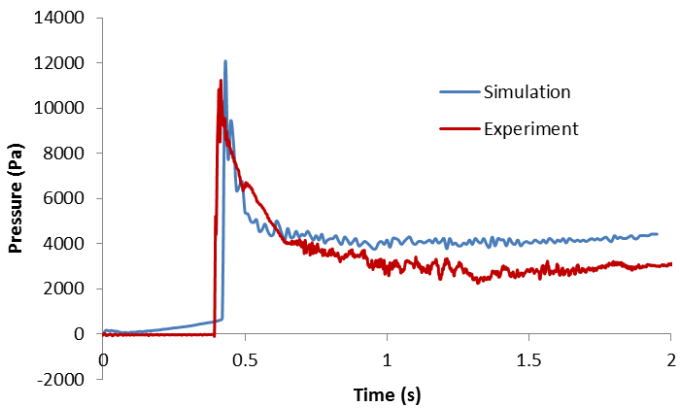
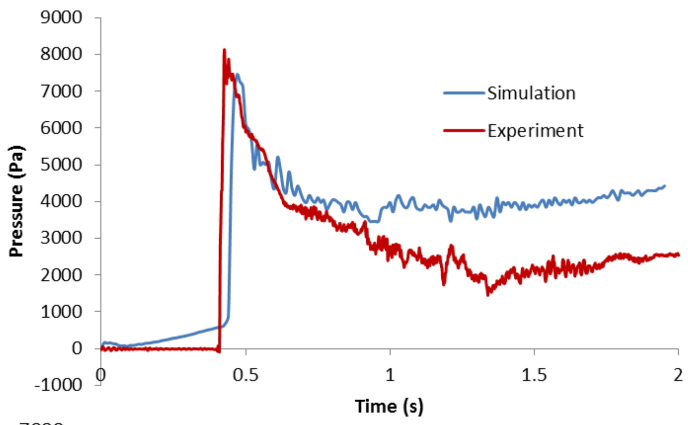


Figure 6.17: Comparison between experimental and multi-phase simulation with the shifting algorithm results for the 4 pressure probes in the side of the obstacle for *dx/h₀*=0.0273

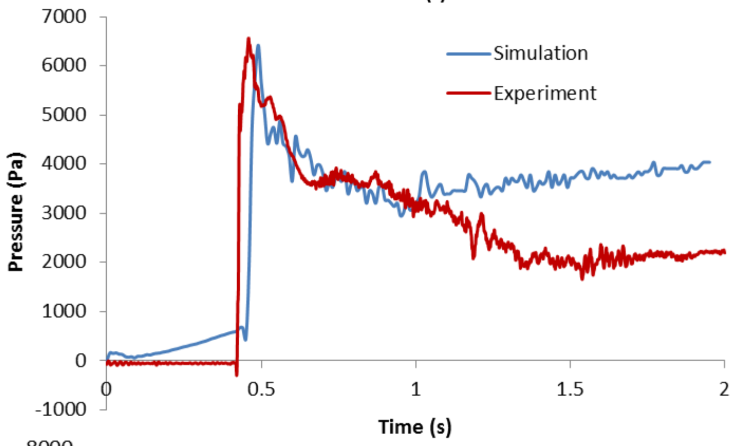
P1



P2



P3



P4

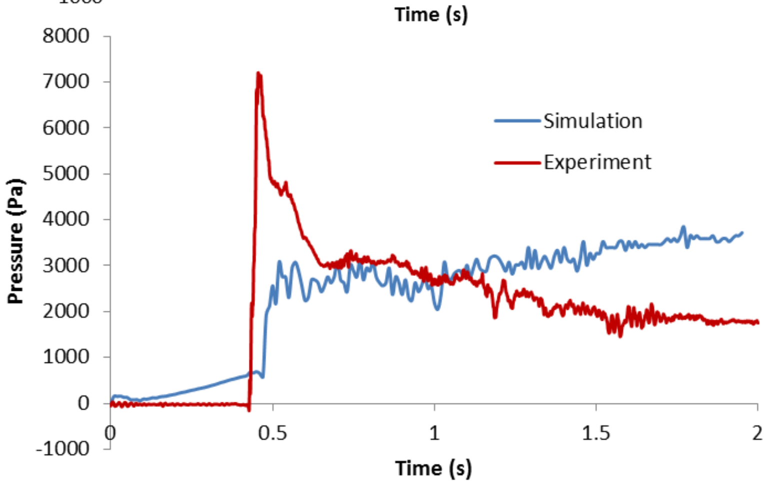


Figure 6.18: Comparison between experimental and a multi-phase simulation with 3.5 million particles without the shifting algorithm for dx/h_0=0.0182

An additional issue is that the pressure peak for probe P4 is not captured regardless of the resolution. Regarding the improvement seen for probe P3 for the high resolution in Figure 6.18 it is possible that the resolution is insufficient to correctly capture the impact at this point. Regarding the pressure probes P5-P8 it was found that they are negatively impacted by the boundaries with the pressure field being oscillatory and constantly increasing. Additional research regarding these two points is required.

6.3. Concluding Remarks

The multi-phase model developed in this study has been extended in the three-dimensional space and validated with experimental and numerical results for the height of the water column and the pressure on the obstacle using 4 million particles. An improvement was shown over corresponding single-phase results. Reasonably good agreement with the experimental results was achieved, with the exception of pressure probe P4, where the pressure peak was not captured. Additional research is also needed regarding the issue of remaining pressure in the domain after the impact.

7. Conclusion

7.1. General Conclusions

This thesis has presented a smoothed particle hydrodynamics (SPH) model using graphics processing units (GPUs) to accelerate the simulation of violent air-water flows in two and three dimensions. The major advantage of using SPH for multi-phase flows is that compared to mesh-based computational methods the highly nonlinear behaviour of the motion of the interface can be implicitly captured with a sharp interface. The robustness and simplicity of the method without the requirement to generate a mesh are other advantages of using SPH for simulating such complex flows.

One major drawback of the SPH method is the large computational resources and time required for complex simulations. This is associated with the number of neighbours per particle and the number of particles, which must be increased in order for the method to be applied for more real applications and complex flows, especially for three-dimensional domains. In this study, GPUs, which have massively parallel capabilities, were selected to provide the necessary acceleration for simulating large particle numbers.

The open source code DualSPHysics, a hybrid CPU-GPU code was heavily modified in order to be able to handle flows with multiple fluids by implementing a multi-phase model that is simple to implement on GPUs (Colagrossi and Landrini, 2003). This choice of formulation enabled different algorithms focusing on identifying an optimised multi-phase SPH algorithm to be investigated for the first time. The use of GPUs enables much greater particle numbers to be used and hence higher resolutions in simulations thereby leading to speedups close to two orders of magnitude over a conventional CPU code. Simulations with millions of particles can now be run in hours meaning that the scheme can be used for design simulation.

The increased resolution enabled by the GPU revealed a problem in the simulations when voids appeared in the gas or air phase. To circumvent this issue a new and improved shifting algorithm was presented for multi-phase SPH schemes. The code was validated by comparison with cases including dry-bed and wet-bed dam break flows in 2-D, a sloshing tank in 2-D and a dry-bed dam break impacting an obstacle in 3-D.

The model should be useful for engineers to investigate multi-phase flows involving violent hydrodynamics such as breaking waves, sloshing and impact. Table 5 shows some general guidelines for using the multi-phase model.

Cohesion Coefficient	Best results are obtained by using a characteristic length with a similar value to the particle spacing.
Background pressure	A small value can smooth the velocity gradient at the interface at the start of the simulation but it is not essential.
Speed of sound	The lighter phase must have a larger value to obtain similar pressure between the two phases in the interface. The value for the heavier phase can be found by the maximum velocity of the simulation. Their ratio depends on the case.
Artificial viscosity	Its importance for stability lessens as the resolution increases. A small value was used in most cases.
Density filter	Needed to remove pressure noise and should be used for each phase separately.
Shifting Algorithm	Needed to enforce Fickian motion in the lighter phase especially in higher resolutions. Improves stability allowing for a decrease in the speed of sound for the lighter phase.
Free-surface correction	Essential in the heavier phase if the shifting algorithm is used

Table 5: General guidelines for using a multi-phase model with SPH

7.2. Detailed Conclusions

7.2.1. Multi-phase algorithms for GPUs

Four different algorithms for optimising the simulation of multi-phase SPH on a GPU were investigated in order to optimise the code. The four algorithms investigated using conditional if-statements, conditional binary multipliers to operate on identical equations within the code, separate particle lists for each phase and an intermediate CPU-GPU function to minimise interaction between the CPU and the GPU. The results from the different algorithms showed that computing the inter-particle forces is the most computationally intensive aspect. Algorithms utilising different particle and neighbour lists for each phase reduced the cost of computing these forces improving the runtime of the simulation.

The improvement was increased as the complexity and the number of particles was increased resulting in over 10% improvement over a conditional *if*-statement algorithm for a three-dimensional domain. A large acceleration of the simulation compared to a conventional CPU-only approach was achieved so that now multi-phase SPH simulations with millions of particles can be performed in a matter of hours on a desktop machine rather than days or months on a supercomputer.

7.2.2. A new particle shifting algorithm for multi-phase SPH schemes

The use of the GPU architecture enabled the modelling of cases with millions of fluid particles. This resulted in some previously unreported problems regarding the simulation of the air phase. Specifically, in high resolutions, voids would appear in the air phase for certain cases, especially if the flow velocity was high or large forces were exerted from the water phase.

The reason for these issues was that the air was being simulated as a highly compressible liquid, lacking the ability to rapidly expand as a gas. To rectify that problem, this thesis has proposed a new particle shifting algorithm for multi-phase flows extending the work of Xu *et al.* (2009) and modified by Lind *et al.*(2012b) and Skillen *et al.* (2013). This algorithm shifts particles towards areas of lower concentration and was able to allow air expansion in a uniform manner, removing the voids.

The effect of the shifting algorithm on the flow and the differences of the particle positions with the simple multi-phase code were investigated. In particular, particle shifting was tested both when only applied to the air phase and when applied to both phases. The free-surface term by Lind *et al.*(2012b) was found to have a significant effect on the evolution of the flow, restricting the expansion of the water phase on the free-surface. It was concluded that the shifting algorithm should be used in the air phase without the free-surface term, while for the water phase either the free-surface term should be used or the shifting algorithm should be removed entirely. Using the shifting algorithm completely prevents the formation of unphysical voids and improves pressure fields.

7.2.3. Validation of the multi-phase scheme

To validate the proposed scheme, a simulation of a dam break experiment with a dry bed was performed. The results for the height of the water column and the toe of the water flow were compared to the experimental results of Koshizuka and Oka (1996) and a corresponding single-phase simulation. Results for the two SPH simulations were similar and in good agreement with the experimental results. Using the shifting algorithm also provided similar results, showing that the initial stages of the dam break are insensitive to the parameters used.

A different dry dam break case by Colagrossi and Landrini (2003) was also compared with the SPH simulation. The effect of the shifting algorithm, the density re-normalisation term and the viscosity model in the evolution of the flow and the pressure field was investigated. The results were compared to the SPH simulation of Colagrossi and Landrini (2003), a Boundary Element Method model by Faltinsen *et al.* (2004) and a Level-Set algorithm by Colicchio *et al.* (2005) for different instances of the flow. Results obtained showed a large numerical viscosity for the lower resolution resulting in a lower plunging wave but they were in good agreement for a higher resolution when the shifting algorithm was used.

The shifting algorithm was found to impact the evolution of the flow altering the shape of the reflected plunging wave and the splash-up. The effect was amplified when the artificial viscosity model was introduced. The shifting algorithm also had an impact on the compressibility of the flow leading to an increased pressure field.

The results of a multi-phase simulation were compared to the wet-bed dam break experiment by Janosi *et al.* (2004) as well as a single-phase simulation. The flow studied also included the movement of the gate. The comparison was performed for two different resolutions and an improved agreement with the experimental results was produced for the multi-phase simulation while an improvement was also shown for the higher resolution.

Sloshing inside a rolling tank was also examined. The case was found to be very sensitive in changes to the viscosity model and the speed of sound of the phases which changes their compressibility. A sensitivity analysis was performed for a range of different values comparing the results to the experiment of Botia-Vera *et al.* (2010). For the lateral impact case, the results obtained show that a good agreement with the experimental results is dependent on the parameters as expected for the formulation of Colagrossi and Landrini (2003).

Finally, a 3-D dry dam break case where the water flow is impacting an obstacle was also studied. The results obtained were compared to the experimental results of Kleefsman *et al.*(2005) and a single-phase SPH simulation by Crespo *et al.* (2011a) for the height of the water flow at different points in the domain. The results obtained were generally in good agreement with the experiment and an improvement on the single-phase model.

Comparison with the experimental results was also performed for the pressure in the side of the obstacle. A very high resolution was found to be needed in order to sufficiently resolve this problem, lower resolutions greatly underestimating the pressure peak. For the higher resolution, the pressure peak was in good agreement for the majority of the pressure probes, however the subsequent pressure decrease was not modelled sufficiently (a problem present in lower resolutions as well).

7.3. Future Work

7.3.1. Alternative boundary conditions

Boundary conditions in SPH have been a well-known issue for some time and have not been the focus of this thesis. However, as seen for the lateral impact case in the sloshing tank in Section 5.5.2 and as mentioned in the dry dam break case in Section 5.2.1, the boundaries are the cause of some significant issues with the SPH simulations. They also have a significant effect on the pressure values which were the focus of this study both for the rolling tank case (Section 5.5) and for the obstacle impact case (Section 6.2). The fictitious boundaries used in this case have two major disadvantages for a violent multi-phase flow: a) they are greatly dependent on fluid particle properties and b) they exert large forces in the fluid particles while not taking into account the flow conditions.

The first issue occurs because of the different properties of the fluids in this study. In order to maintain a stable field, the properties of the boundaries were approximated as the heavier fluid. The forces exerted on the lighter fluid are then increased leading to a separation between the boundary and the fluid and pressure and density oscillation for the air particles as seen in Section 5.5.3. The second issue appears in the obstacle impact but can be observed more clearly for the dry dam break case in Section 5.2.1. In this case the force exerted from the boundaries causes a delay in the movement of the water particles resulting in discrepancies with the experimental results. The boundaries also affect the reflected wave as seen in Section 5.3.5.

To resolve this issue an alternative approach should be selected. A suitable model should either not be dependent on particle properties or change depending on the fluid particle it is simulating. Possible approaches include a semi-analytical boundary approach (Ferrand *et al.*, 2010) or a zero and first-order consistent boundary condition (Fourtakas *et al.*, 2013b). At this point, both require significant investment to be applied in a multi-phase GPU code, the first requiring resolving issues in the interface and the second requiring an extension to 3-D space.

7.3.2. Alternative Multi-phase Model

The multi-phase model of Colagrossi and Landrini (2003) is a very simple algorithm which makes it ideal for applying it to a GPU-based code. Its simplicity however, is derived from its dependence on a number of numerical coefficients defined by the user to determine the interaction between the fluids and this arbitrary choice is a deterrent for more widespread engineering applications of SPH. For example, as seen in the rolling tank case in Section 5.5 the pressure peak was greatly dependent on the numerical parameters of the model.

More importantly, the model revealed issues with the air modelling in high resolutions with the shifting algorithm being used to eliminate them and the air phase is not being modelled as a gas but as a highly compressible liquid. A different issue with the model is the increased computational time it requires. The speed of sound for the air needs to be sufficiently larger than the water causing a decrease on the time step which depends on the maximum speed of sound of the flow.

As mentioned in Section 2.4 a number of alternative models have been developed for multi-phase flows. Of particular interest is the model of Leduc *et al.* (2009) who used a Riemann solver to model the interface between the two phases and the model of Grenier *et al.*(2009) who combine the Colagrossi and Landrini (2003) model with a specific volume approach and a new density renormalisation term. The first case eliminates the need for numerical coefficients to maintain the cohesion of the flow while the second allows for a smaller value of the speed of sound to be used for the air phase, therefore decreasing the computational runtime.

A feature lacking from the current model is the ability to model the surface tension forces. For the cases investigated herein, the surface tension forces did not affect the results; however, in any case involving bubble flow, the surface tension forces are pivotal in accurately simulating the creation, separation and behaviour of the bubbles. Introducing a surface tension model, such as the continuum surface force (Brackbill *et al.*, 1992) will enable the scheme to be applied to a range of new cases.

7.3.3. Further Application of the model

The scheme created in this study is sufficiently robust to be used for the investigation of a variety of multi-phase flows. Of particular interest is the study of breaking waves and especially plunging breakers, which are a highly transient flow with constantly changing free-surface. A preliminary investigation is being performed on the behaviour of solitary waves breaking on a slope.

A case where the behaviour of the air has a large impact in the eventual force distribution is slam modelling. As shown recently by Lind *et al.* (2013) when modelling the impact of a flat plate on a water surface, trapped air has a cushioning effect, reducing the local pressures and impact forces generated. The instantaneous nature of the impact and the small volume of the trapped air greatly increase the computational cost of this case necessitating the use of a large number of particles, issues which could be addressed by the hardware acceleration employed in this scheme

In conjunction with the surface tension model mentioned in the previous section, bubble flow and separation as well as the effect of aeration could also be modelled using this scheme. The incorporation of periodic boundaries in the latest version of DualSPHysics will also allow the modelling of partially-filled pipe flows.

7.3.4. Creation of a multi-GPU code

The use of GPUs as a means to accelerate an SPH code can significantly decrease the computational runtime, as shown in this study. However, GPUs are still limited in two ways: (a) the simulations performed here are only executed with single precision; (b) the amount of memory in a single GPU is limited. The severity of both issues is dependent on the hardware used, but regardless they limit the resolution and the number of particles that can be processed.

To circumvent these limits a solution is the connection of several GPUs using the MPI protocol with each GPU solving the flow for only a portion of these particles. A recently released version of the DualSPHysics code has already included the MPI protocol and has already been used to perform a simulation with 1 billion particles (Dominguez *et al.*, 2013a). Transferring the multi-phase model to this version is an entirely viable possibility.

References

AMADA, T., IMURA, M., YASUMURO, M., MANABE, Y. & CHIHARA, K. 2004. Particle-based fluid simulation on the GPU. *ACM Workshop on General-purpose Computing on Graphics Processors.* Los Angeles.

AMDAHL, G. 1967. The validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Spring Joint Computer Conference.* Atlantic City,New Jersey.

ATLURI, S. N. & SHEN, S. P. 2002. The meshless local Petrov-Galerkin (MLPG) method: A simple & less-costly alternative to the finite element and boundary element methods. *Cmes-Computer Modeling in Engineering & Sciences,* 3**,** 11-51.

ATLURI, S. N. & ZHU, T. 1998. A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics,* 22**,** 117-127.

ATLURI, S. N. & ZHU, T. L. 2000. The meshless local Petrov-Galerkin (MLPG) approach for solving problems in elasto-statics. *Computational Mechanics,* 25**,** 169-179.

BALSARA, D. S. 1995. VON-NEUMANN STABILITY ANALYSIS OF SMOOTHED PARTICLE HYDRODYNAMICS - SUGGESTIONS FOR OPTIMAL-ALGORITHMS. *Journal of Computational Physics,* 121**,** 357-372.

BARBA, L. A., LEONARD, A. & ALLEN, C. B. 2005. Advances in viscous vortex methods - meshless spatial adaption based on radial basis function interpolation. *International Journal for Numerical Methods in Fluids,* 47**,** 387-421.

BARREIRO, A., CRESPO, A. J. C., DOMINGUEZ, J. M. & GOMEZ-GESTEIRA, M. 2013. Smoothed Particle Hydrodynamics for coastal engineering problems. *Computers & Structures,* 120**,** 96-106.

BASS, R. L., HOKANSON, J. C. & COX, P. A. 1976. Verification of LNG tank loading criteria. *Ship Structure Committee Report.* Ship Structure Committee.

BATCHELOR, G. K. 1967. *An introduction to fluid dynamics,* Cambridge, UK, Cambridge University Press.

BELYTSCHKO, T., KRONGAUZ, Y., DOLBOW, J. & GERLACH, C. 1998. On the completeness of meshfree particle methods. *International Journal for Numerical Methods in Engineering,* 43**,** 785-+.

BELYTSCHKO, T., KRONGAUZ, Y., ORGAN, D., FLEMING, M. & KRYSL, P. 1996. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering,* 139**,** 3-47.

BELYTSCHKO, T., LU, Y. Y. & GU, L. 1994. ELEMENT-FREE GALERKIN METHODS. *International Journal for Numerical Methods in Engineering,* 37**,** 229-256.

BELYTSCHKO, T., LU, Y. Y. & GU, L. 1995. CRACK-PROPAGATION BY ELEMENT-FREE GALERKIN METHODS. *Engineering Fracture Mechanics,* 51**,** 295-315.

BENZ, W. & ASPHAUG, E. 1994. IMPACT SIMULATIONS WITH FRACTURE .1. METHOD AND TESTS. *Icarus,* 107**,** 98-116.

BENZ, W. & ASPHAUG, E. 1995. SIMULATIONS OF BRITTLE SOLIDS USING SMOOTH PARTICLE HYDRODYNAMICS. *Computer Physics Communications,* 87**,** 253-265.

BIN, A. K. 1993. GAS ENTRAINMENT BY PLUNGING LIQUID JETS. *Chemical Engineering Science,* 48**,** 3585-3630.

BOEK, E. S., COVENEY, P. V., LEKKERKERKER, H. N. W. & VANDERSCHOOT, P. 1997. Simulating the rheology of dense colloidal suspensions using dissipative particle dynamics. *Physical Review E,* 55**,** 3124-3133.

BONET, J. & KULASEGARAM, S. 2000. Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations. *International Journal for Numerical Methods in Engineering,* 47**,** 1189-1214.

BONET, J. & LOK, T. S. L. 1999. Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering,* 180**,** 97-115.

BORVE, S. 2011. Generalised ghost particle method for handling reflecting boundaries. *6th International SPHERIC Workshop.* Hamburg, Germany.

BOTIA-VERA, E., SOUTO-IGLESIAS, A., BULIAN, G. & LOBOVSKY, L. 2010. Three SPH novel benchmark test cases for free surface flows. *5th ERCOFTAC SPHERIC Workshop.* Manchester, UK.

BRACKBILL, J. U., KOTHE, D. B. & ZEMACH, C. 1992. A CONTINUUM METHOD FOR MODELING SURFACE-TENSION. *Journal of Computational Physics,* 100**,** 335-354.

BREDMOSE, H., HUNT-RABY, A., JAYARATNE, R. & BULLOCK, G. N. 2010. The ideal flip-through impact: experimental and numerical investigation. *Journal of Engineering Mathematics,* 67**,** 115-136.

BROOKSHAW, L. 1985. A METHOD OF CALCULATING RADIATIVE HEAT DIFFUSION IN PARTICLE SIMULATIONS. *Proceedings Astronomical Society of Australia,* 6**,** 207-210.

BUCHNER, B. 1996. The impact of green water on FPSO design. *Offshore Technology Conference.*

BULLOCK, G., OBHRAI, C., MULLER, G., WOLTERS, G., PEREGRINE, D. H. & BREDMOSE, H. 2004. Characteristics and design implications of breaking wave impacts. *In:* MCKEE-SMITH, J. (ed.) *29th Interanational Conference of Coastal Engineering.* ASCE.

BUTCHER, J. C. 1964. IMPLICIT RUNGE-KUTTA PROCESSES. *Mathematics of Computation,* 18**,** 50-&.

BUTCHER, J. C. 2003. *Numerical Methods for Ordinary Differential Equations,* New York, John Wiley & Sons.

CHA, S. H. & WHITWORTH, A. P. 2003. Implementations and tests of Godunov-type particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society,* 340**,** 73-90.

CHEN, J. S., PAN, C., ROQUE, C. & WANG, H. P. 1998. A Lagrangian reproducing kernel particle method for metal forming analysis. *Computational Mechanics,* 22**,** 289-307.

CHEN, J. S., PAN, C. & WU, C. T. 1997. Large deformation analysis of rubber based on a reproducing kernel particle method. *Computational Mechanics,* 19**,** 211-227.

CHEN, J. S., WU, C. T., YOON, S. & YOU, Y. 2001. A stabilized conforming nodal integration for Galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering,* 50**,** 435-466.

CHORIN, A. J. 1968. NUMERICAL SOLUTION OF NAVIER-STOKES EQUATIONS. *Mathematics of Computation,* 22**,** 745-&.

CHORIN, A. J. 1973. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics,* 57**,** 785-796.

CHORIN, A. J. 1978. VORTEX SHEET APPROXIMATION OF BOUNDARY-LAYERS. *Journal of Computational Physics,* 27**,** 428-442.

CLARK, D. 1998. OpenMP: a parallel standard for the masses. *Ieee Concurrency,* 6**,** 10-12.

COLAGROSSI, A., ANTUONO, M. & MARRONE, S. 2009. A 2D+t SPH model with enhanced solid boundary treatment. *4th ERCOFTAC SPHERIC Workshop.* Nantes, France.

COLAGROSSI, A. & LANDRINI, M. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics,* 191**,** 448-475.

COLICCHIO, G., LANDRINI, M. & CHAPLIN, J. R. 2005. Level-set computations of free surface rotational flows. *Journal of Fluids Engineering-Transactions of the Asme,* 127**,** 1111-1121.

COTTET, G. H., KOUMOUTSAKOS, P. & SALIHI, M. L. O. 2000. Vortex methods with spatially varying cores. *Journal of Computational Physics,* 162**,** 164-185.

CRESPO, A. C., DOMINGUEZ, J. M., BARREIRO, A., GOMEZ-GESTEIRA, M. & ROGERS, B. D. 2011a. GPUs, a New Tool of Acceleration in CFD: Efficiency and Reliability on Smoothed Particle Hydrodynamics Methods. *Plos One,* 6.

CRESPO, A. J. C., DOMINGUEZ, J. M., BARREIRO, A. & GOMEZ-GESTEIRA, M. 2010. Development of a dual CPU-GPU SPH model. *5th International SPHERIC Workshop.* Manchester.

CRESPO, A. J. C., DOMINGUEZ, J. M., BARREIRO, A., GOMEZ-GESTEIRA, M. & ROGERS, B. D. 2011b. DualSPHysics, new GPU computing on SPH models. *6th ERCOFTAC SPHERIC Workshop.* Hamburg, Germany.

CRESPO, A. J. C., GOMEZ-GESTEIRA, M. & DALRYMPLE, R. A. 2007. Boundary conditions generated by dynamic particles in SPH methods. *Cmc-Computers Materials & Continua,* 5**,** 173-184.

CRESPO, A. J. C., GOMEZ-GESTEIRA, M. & DALRYMPLE, R. A. 2008. Modeling Dam Break Behavior over a Wet Bed by a SPH Technique. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 134**,** 313-320.

CRESPO, A. J. C., MARONGIU, J. C., PARKINSON, E., GOMEZ-GESTEIRA, M. & DOMINGUEZ, J. M. 2009. High performance of SPH codes: Best approaches for efficient parallelisation on GPU computing. *4th ERCOFTAC SPHERIC Workshop.* Nantes, France.

CUMMINS, S. J. & RUDMAN, M. 1999. An SPH projection method. *Journal of Computational Physics,* 152**,** 584-607.

DAGUM, L. & MENON, R. 1998. OpenMP: An industry standard API for shared-memory programming. *Ieee Computational Science & Engineering,* 5**,** 46-55.

DALRYMPLE, R. A. & ROGERS, B. D. 2006. Numerical modeling of water waves with the SPH method. *Coastal Engineering,* 53**,** 141-147.

DE LEFFE, M., LE TOUZE, D. & ALESSANDRINI, B. 2009. Normal flux at the boundary for SPH. *4th ERCOFTAC SPHERIC Workshop.* Nantes, France.

DE PADOVA, D., MOSSA, M., SIBILLA, S. & TORTI, E. 2010. Hydraulic jump simulation by SPH. *5th International SPHERIC Workshop.* Manchester, UK.

DI MONACO, A., MANENTI, S., GALLATI, M., SIBILLA, S., AGATE, G., GUANDALINI, R. & MAFFIO, A. 2009. A semi-analytic approach for SPH modelling of solid boundaries. *4th International SPHERIC Workshop.* Nantes, France.

DOMINGUEZ, J. M., CRESPO, A. J. C., GOMEZ-GESTEIRA, M. & MARONGIU, J. C. 2010. Neighbour Lists in Smoothed Particle Hydrodynamics. *International Journal for Numerical Methods in Fluids*.

DOMINGUEZ, J. M., CRESPO, A. J. C., GOMEZ-GESTEIRA, M. & MARONGIU, J. C. 2011. Neighbour lists in smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids,* 67**,** 2026-2042.

DOMINGUEZ, J. M., CRESPO, A. J. C., GOMEZ-GESTEIRA, M. & ROGERS, B. D. 2013a. Simulating more than 1 billion SPH particles using GPU hardware acceleration. *8th International SPHERIC Workshop.* Trondheim, Norway.

DOMINGUEZ, J. M., CRESPO, A. J. C., VALDEZ-BALDERAS, D., ROGERS, B. D. & GOMEZ-GESTEIRA, M. 2013b. New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Computer Physics Communications,* 184**,** 1848-1860.

DULLWEBER, A., LEIMKUHLER, B. & MCLACHLAN, R. 1997. Symplectic splitting methods for rigid body molecular dynamics. *Journal of Chemical Physics,* 107**,** 5840-5851.

DYKA, C. T. & INGEL, R. P. 1995. AN APPROACH FOR TENSION INSTABILITY IN SMOOTHED PARTICLE HYDRODYNAMICS (SPH). *Computers & Structures,* 57**,** 573-580.

ELLERO, M., SERRANO, M. & ESPANOL, P. 2007. Incompressible smoothed particle hydrodynamics. *Journal of Computational Physics,* 226**,** 1731-1752.

ESPANOL, P. 1995. HYDRODYNAMICS FROM DISSIPATIVE PARTICLE DYNAMICS. *Physical Review E,* 52**,** 1734-1742.

ESPANOL, P. & WARREN, P. 1995. STATISTICAL-MECHANICS OF DISSIPATIVE PARTICLE DYNAMICS. *Europhysics Letters,* 30**,** 191-196.

FALTINSEN, O. M., LANDRINI, M. & GRECO, M. 2004. Slamming in marine applications. *Journal of Engineering Mathematics,* 48**,** 187-217.

FALTINSEN, O. M., ROGNEBAKKE, O. F., LUKOVSKY, I. A. & TIMOKHA, A. N. 2000. Multidimensional modal analysis of nonlinear sloshing in a rectangular tank with finite water depth. *Journal of Fluid Mechanics,* 407**,** 201-234.

FELDMAN, J. & BONET, J. 2007. Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering,* 72**,** 295-324.

FERRAND, M., LAURENCE, D., ROGERS, B. D. & VIOLEAU, D. 2010. Improved time-scheme integration approach for dealing with semi-analytical boundary conditions in SPARTACUS2D. *5th SPHERIC International Workshop.* Manchester, UK.

FERRAND, M., LAURENCE, D. R., ROGERS, B. D., VIOLEAU, D. & KASSIOTIS, C. 2013. Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids,* 71**,** 446-472.

FERRARI, A., DUMBSER, M., TORO, E. F. & ARMANINI, A. 2009. A new 3D parallel SPH scheme for free surface flows. *Computers & Fluids,* 38**,** 1203-1217.

FLEKKOY, E. G., COVENEY, P. V. & DE FABRITIIS, G. 2000. Foundations of dissipative particle dynamics. *Physical Review E,* 62**,** 2140-2157.

FLEMING, M., CHU, Y. A., MORAN, B. & BELYTSCHKO, T. 1997. Enriched element-free Galerkin methods for crack tip fields. *International Journal for Numerical Methods in Engineering,* 40**,** 1483-1504.

FOURTAKAS, G., ROGERS, B. D. & LAURENCE, D. R. 2013a. Modelling sediment resuspension in industrial tanks using SPH. *Houille Blanche-Revue Internationale De L Eau***,** 39-45.

FOURTAKAS, G., VACONDIO, R. & ROGERS, B. D. 2013b. SPH approximate zeroth and first-order consistent boundary conditions for irregular boundaries. *8th International SPHERIC Workshop.* Trondheim, Norway.

GINGOLD, R. A. & MONAGHAN, J. J. 1977. SMOOTHED PARTICLE HYDRODYNAMICS - THEORY AND APPLICATION TO NON-SPHERICAL STARS. *Monthly Notices of the Royal Astronomical Society,* 181**,** 375-389.

GINGOLD, R. A. & MONAGHAN, J. J. 1982. KERNEL ESTIMATES AS A BASIS FOR GENERAL PARTICLE METHODS IN HYDRODYNAMICS. *Journal of Computational Physics,* 46**,** 429-453.

GOMEZ-GESTEIRA, M. & DALRYMPLE, R. A. 2004. Using a three-dimensional smoothed particle hydrodynamics method for wave impact on a tall structure. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 130**,** 63-69.

GOMEZ-GESTEIRA, M., ROGERS, B. D., CRESPO, A. J. C., DALRYMPLE, R. A., NARAYANASWAMY, M. & DOMINGUEZ, J. M. 2012. SPHysics - development of a free-surface fluid solver - Part 1: Theory and formulations. *Computers & Geosciences,* 48**,** 289-299.

GOMEZ-GESTEIRA, M., ROGERS, B. D., DALRYMPLE, R. A. & CRESPO, A. J. C. 2010. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research,* 48**,** 6-27.

GORF, P., BARLTROP, N., OKAN, B., HODSON, T. & RAINEY, R. FPSO bow damage in steep waves. Rogue Waves, 2000 Brest.

GRAY, J. P., MONAGHAN, J. J. & SWIFT, R. P. 2001. SPH elastic dynamics. *Computer Methods in Applied Mechanics and Engineering,* 190**,** 6641-6662.

GREENGARD, L. & ROKHLIN, V. 1987. A FAST ALGORITHM FOR PARTICLE SIMULATIONS. *Journal of Computational Physics,* 73**,** 325-348.

GRENIER, N., ANTUONO, M., COLAGROSSI, A., LE TOUZE, D. & ALESSANDRINI, B. 2009. An Hamiltonian interface SPH formulation for multi-fluid and free surface flows. *Journal of Computational Physics,* 228**,** 8380-8393.

GROPP, W., LUSK, E. & SKJELLUM, A. 1994. *Using MPI: Portable parallel programming with the message-parsing interface,* Cambridge, Massachusetts, USA, MIT Press.

GUIFFAUT, C. & MAHDJOUBI, K. 2001. A parallel FDTD algorithm using the MPI library. *Ieee Antennas and Propagation Magazine,* 43**,** 94-103.

GUO, X., LIND, S., ROGERS, B. D., STANSBY, P. K. & ASHWORTH, M. 2013. Efficient massive parallelisation for incompressible Smoothed Particle Hydrodynamics with 10^8 particles. *6th International SPHERIC Workshop.* Trondheim, Norway.

HARADA, T., KOSHIZUKA, S. & KAWAGUCHI, Y. 2007a. Sliced Data Structure for Particle-Based Simulations on GPUs. *Graphite 2007: 5th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southern Asia, Proceedings***,** 55-62.

HARADA, T., KOSHIZUKA, S. & KAWAGUCHI, Y. 2007b. Smoothed Particle Hydrodynamics on GPUs. *Proc. of Computer Graphics International (2007), pp. 63-70*.

HERAULT, A., BILOTTA, G. & DALRYMPLE, R. A. 2010. SPH on GPU with CUDA. *Journal of Hydraulic Research,* 48**,** 74-79.

HEWITT, G. F. 1982. *In:* HETSRONI, G. (ed.) *Handbook of multi-phase systems.* New York, USA: Hemisphere Publishing Corporation.

HOLTHUIJSEN, L. H. 2007. *Waves in oceanic and coastal waters,* New York, USA, Cambridge University Press.

HOOGERBRUGGE, P. J. & KOELMAN, J. 1992. SIMULATING MICROSCOPIC HYDRODYNAMIC PHENOMENA WITH DISSIPATIVE PARTICLE DYNAMICS. *Europhysics Letters,* 19**,** 155-160.

HOOVER, W. G. 1998. Isomorphism linking smooth particles and embedded atoms. *Physica A,* 260**,** 244-254.

HU, K., MINGHAM, C. G. & CAUSON, D. M. 2000. Numerical simulation of wave overtopping of coastal structures using the non-linear shallow water equations. *Coastal Engineering,* 41**,** 433-465.

HU, X. Y. & ADAMS, N. A. 2006. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics,* 213**,** 844-861.

HU, X. Y. & ADAMS, N. A. 2007. An incompressible multi-phase SPH method. *Journal of Computational Physics,* 227**,** 264-278.

HU, X. Y. & ADAMS, N. A. 2009. A constant-density approach for incompressible multi-phase SPH. *Journal of Computational Physics,* 228**,** 2082-2091.

HUBBARD, M. E. & DODD, N. 2002. A 2D numerical model of wave run-up and overtopping. *Coastal Engineering,* 47**,** 1-26.

ISSA, R. 2005. *Numerical assessment of the Smoothed Particle Hydrodynamics gridless method for incompressible flows and its extension to turbulent flows.* PhD, UMIST.

ISSA, R. & VIOLEAU, D. 2009. Modelling breaking solitary waves with eddy-viscosity turbulent SPH models. *Comput. Mater. Continua,* 8**,** 151-164.

ISSA, R., VIOLEAU, D., LEE, E. S. & FLAMENT, H. 2009. Modelling nonlinear water waves with RANS and LES SPH models. *In:* MA, Q. W. (ed.) *Advances in Numerical Simulation of Nonlinear Water Waves.* London: World Scientific Publishing Co.

JANOSI, I. M., JAN, D., SZABO, K. G. & TEL, T. 2004. Turbulent drag reduction in dam-break flows. *Experiments in Fluids,* 37**,** 219-229.

JUN, S., LIU, W. K. & BELYTSCHKO, T. 1998. Explicit reproducing kernel particle methods for large deformation problems. *International Journal for Numerical Methods in Engineering,* 41**,** 137-166.

KAHAN, W. 1965. FURTHER REMARKS ON REDUCING TRUNCATION ERRORS. *Communications of the Acm,* 8**,** 40-&.

KATZ, N. 1992. DISSIPATIONAL GALAXY FORMATION .2. EFFECTS OF STAR FORMATION. *Astrophysical Journal,* 391**,** 502-517.

KHAYYER, A. & GOTOH, H. 2008. DEVELOPMENT OF CMPS METHOD FOR ACCURATE WATER-SURFACE TRACKING IN BREAKING WAVES. *Coastal Engineering Journal,* 50**,** 179-207.

KHAYYER, A. & GOTOH, H. 2010a. A higher order Laplacian model for enhancement and stabilization of pressure calculation by the MPS method. *Applied Ocean Research,* 32**,** 124-131.

KHAYYER, A. & GOTOH, H. 2010b. On particle-based simulation of a dam break over a wet bed. *Journal of Hydraulic Research,* 48**,** 238-249.

KHAYYER, A., GOTOH, H. & SHAO, S. D. 2008. Corrected Incompressible SPH method for accurate water-surface tracking in breaking waves. *Coastal Engineering,* 55**,** 236-250.

KLEEFSMAN, K. M. T., FEKKEN, G., VELDMAN, A. E. P., IWANOWSKI, B. & BUCHNER, B. 2005. A Volume-of-Fluid based simulation method for wave impact problems. *Journal of Computational Physics,* 206**,** 363-393.

KOBAYASHI, N. 1997. Wave runup and overtopping on beaches and coastal structures. Center of Applied Ocean Research.

KOLB, A. & CUNTZ, N. 2005. Dynamic particle coupling for GPU-based fluid simulation. *18th Symposium on Simulation Technique.* Erlangen, Germany.

KOSHIZUKA, S., NOBE, A. & OKA, Y. 1998. Numerical analysis of breaking waves using the moving particle semi-implicit method. *International Journal for Numerical Methods in Fluids,* 26**,** 751-769.

KOSHIZUKA, S. & OKA, Y. 1996. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Science and Engineering,* 123**,** 421-434.

KOUMOUTSAKOS, P. 2005. Multiscale flow simulations using particles. *Annual Review of Fluid Mechanics,* 37**,** 457-487.

KOUMOUTSAKOS, P. & LEONARD, A. 1995. HIGH-RESOLUTION SIMULATIONS OF THE FLOW AROUND AN IMPULSIVELY STARTED CYLINDER USING VORTEX METHODS. *Journal of Fluid Mechanics,* 296**,** 1-38.

KRYSL, P. & BELYTSCHKO, T. 1995. Analysis of thin plates by the element-free Galerkin method. *Computational Mechanics,* 17**,** 26-35.

KULASEGARAM, S., BONET, J., LEWIS, R. W. & PROFIT, M. 2004. A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications. *Computational Mechanics,* 33**,** 316-325.

LANCASTER, P. & SALKAUSKAS, K. 1981. SURFACES GENERATED BY MOVING LEAST-SQUARES METHODS. *Mathematics of Computation,* 37**,** 141-158.

LE TOUZÉ, D., COLAGROSSI, A. & COLICCHIO, G. 2006. Ghost technique for squared angles applied to the solution of benchmarks 1 and 2. *1st International SPHERIC Workshop.* Rome, Italy.

LEDUC, J., LEBOEUF, F., LANCE, M., PARKINSON, E. & MARONGIU, J. C. 2010. Improvement of multiphase model using preconditioned Riemann solvers. *5th International SPHERIC Workshop.* Manchester, UK.

LEDUC, J., MARONGIU, J. C., LEBOEUF, F., LANCE, M. & PARKINSON, E. 2009. Mutliphase SPH: A new model based on acoustic Riemann solver. *4th ERCOFTAC SPHERIC Workshop.* Nantes, France.

LEE, E.-S., VIOLEAU, D., ISSA, R. & PLOIX, S. 2010. Application of weakly compressible and truly incompressible SPH to 3-D water collapse in waterworks. *Journal of Hydraulic Research,* 48**,** 50-60.

LEE, E. S., MOULINEC, C., XU, R., VIOLEAU, D., LAURENCE, D. & STANSBY, P. 2008. Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. *Journal of Computational Physics,* 227**,** 8417-8436.

LEONARDI, M., MANENTI, S. & SIBILA, S. 2011. 2D and 3D sloshing with SPH. *7th International SPHERIC Workshop.* Prato, Italy.

LI, Y. & RAICHLEN, F. 2002. Non-breaking and breaking solitary wave run-up. *Journal of Fluid Mechanics,* 456**,** 295-318.

LIBERSKY, L. D., PETSCHEK, A. G., CARNEY, T. C., HIPP, J. R. & ALLAHDADI, F. A. 1993. HIGH-STRAIN LAGRANGIAN HYDRODYNAMICS - A 3-DIMENSIONAL SPH CODE FOR DYNAMIC MATERIAL RESPONSE. *Journal of Computational Physics,* 109**,** 67-75.

LIN, H. & ATLURI, S. N. 2000. Meshless local Petrov-Galerkin (MLPG) method for convection-diffusion problems. *Cmes-Computer Modeling in Engineering & Sciences,* 1**,** 45-60.

LIN, H. & ATLURI, S. N. 2001. The Meshless Local Petrov-Galerkin (MLPG) method for solving incompressible Navier-Stokes equations. *Cmes-Computer Modeling in Engineering & Sciences,* 2**,** 117-142.

LIN, P. Z., CHANG, K. A. & LIU, P. L. F. 1999. Runup and rundown of solitary waves on sloping beaches. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 125**,** 247-255.

LIND, S., STANSBY, P. K. & ROGERS, B. D. 2012a. A multiphase compressible-incompressible smmotthe particle hydrodynamics method. *7th International SPHERIC Workshop.* Hamburg, Germany.

LIND, S. J., STANSBY, P. K. & ROGERS, B. D. 2013. Slam modelling with SPH: the importance of air. *8th International SPHERIC Workshop.* Trondheim, Norway.

LIND, S. J., XU, R., STANSBY, P. K. & ROGERS, B. D. 2012b. Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. *Journal of Computational Physics,* 231**,** 1499-1523.

LIU, G. R. & LIU, M. B. 2003. *Smooth Particle Hydrodynamics - A meshfree particle method*, World Scientific Publishing Co. Pte. Ltd.

LIU, W. K. & CHEN, Y. J. 1995. WAVELET AND MULTIPLE SCALE REPRODUCING KERNEL METHODS. *International Journal for Numerical Methods in Fluids,* 21**,** 901-931.

LIU, W. K., JUN, S., LI, S. F., ADEE, J. & BELYTSCHKO, T. 1995a. REPRODUCING KERNEL PARTICLE METHODS FOR STRUCTURAL DYNAMICS. *International Journal for Numerical Methods in Engineering,* 38**,** 1655-1679.

LIU, W. K., JUN, S. & ZHANG, Y. F. 1995b. REPRODUCING KERNEL PARTICLE METHODS. *International Journal for Numerical Methods in Fluids,* 20**,** 1081-1106.

LIU, W. K., LI, S. F. & BELYTSCHKO, T. 1997. Moving least-square reproducing kernel methods .1. Methodology and convergence. *Computer Methods in Applied Mechanics and Engineering,* 143**,** 113-154.

LONGSHAW, S. 2013. *RE: Implementation of the Kahan Summation Algorithm.* Type to MOKOS, A.

LU, Y. Y., BELYTSCHKO, T. & GU, L. 1994. A NEW IMPLEMENTATION OF THE ELEMENT FREE GALERKIN METHOD. *Computer Methods in Applied Mechanics and Engineering,* 113**,** 397-414.

LUCY, L. B. 1977. NUMERICAL APPROACH TO TESTING OF FISSION HYPOTHESIS. *Astronomical Journal,* 82**,** 1013-1024.

LUGNI, C., BROCCHINI, M. & FALTINSEN, O. M. 2006. Wave impact loads: The role of the flip-through. *Physics of Fluids,* 18.

MARONGIU, J.-C., LEBOEUF, F., CARO, J. & PARKINSON, E. 2010. Free surface flows simulations in Pelton turbines using an hybrid SPH-ALE method. *Journal of Hydraulic Research,* 48**,** 40-49.

MARONGIU, J. C., LEBOEUF, F. & PARKINSON, E. 2007. Numerical simulation of the flow in a Pelton turbine using the meshless method smoothed particle hydrodynamics: a new simple solid boundary treatment. *Proceedings of the Institution of Mechanical Engineers Part a-Journal of Power and Energy,* 221**,** 849-856.

MARSHALL, J. S. & GRANT, J. R. 1996. Penetration of a blade into a vortex core: Vorticity response and unsteady blade forces. *Journal of Fluid Mechanics,* 306**,** 83-109.

MARTIN, J. C. & MOYCE, W. J. 1952. AN EXPERIMENTAL STUDY OF THE COLLAPSE OF LIQUID COLUMNS ON A RIGID HORIZONTAL PLANE .4. *Philosophical Transactions of the Royal Society of London Series a-Mathematical and Physical Sciences,* 244**,** 312-324.

MASSELINK, G. & PULEO, J. A. 2006. Swash-zone morphodynamics. *Continental Shelf Research,* 26**,** 661-680.

MCINTOSH-SMITH, S., WILSON, T., IBARRA, A. A., CRISP, J. & SESSIONS, R. B. 2012. Benchmarking Energy Efficiency, Power Costs and Carbon Emissions on Heterogeneous Systems. *Computer Journal,* 55**,** 192-205.

MONAGHAN, J. J. 1989. ON THE PROBLEM OF PENETRATION IN PARTICLE METHODS. *Journal of Computational Physics,* 82**,** 1-15.

MONAGHAN, J. J. 1992. SMOOTHED PARTICLE HYDRODYNAMICS. *Annual Review of Astronomy and Astrophysics,* 30**,** 543-574.

MONAGHAN, J. J. 1994. SIMULATING FREE-SURFACE FLOWS WITH SPH. *Journal of Computational Physics,* 110**,** 399-406.

MONAGHAN, J. J. 1997. SPH and Riemann solvers. *Journal of Computational Physics,* 136**,** 298-307.

MONAGHAN, J. J. 2000. SPH without a tensile instability. *Journal of Computational Physics,* 159**,** 290-311.

MONAGHAN, J. J. 2005. Smoothed particle hydrodynamics. *Reports on Progress in Physics,* 68**,** 1703-1759.

MONAGHAN, J. J. 2011. A simple and efficient algorithm for multi-fluids. *6th International SPHERIC Workshop.* Hamburg, Germany.

MONAGHAN, J. J., CAS, R. A. F., KOS, A. M. & HALLWORTH, M. 1999. Gravity currents descending a ramp in a stratified tank. *Journal of Fluid Mechanics,* 379**,** 39-70.

MONAGHAN, J. J. & GINGOLD, R. A. 1983. SHOCK SIMULATION BY THE PARTICLE METHOD SPH. *Journal of Computational Physics,* 52**,** 374-389.

MONAGHAN, J. J. & KAJTAR, J. B. 2009a. SPH boundary forces. *4th International SPHERIC Workshop.* Nantes, France.

MONAGHAN, J. J. & KAJTAR, J. B. 2009b. SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications,* 180**,** 1811-1820.

MONAGHAN, J. J. & KOCHARYAN, A. 1995. SPH SIMULATION OF MULTIPHASE FLOW. *Computer Physics Communications,* 87**,** 225-235.

MONAGHAN, J. J. & KOS, A. 1999. Solitary waves on a Cretan beach. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 125**,** 145-154.

MONAGHAN, J. J. & PONGRACIC, H. 1985. ARTIFICIAL VISCOSITY FOR PARTICLE METHODS. *Applied Numerical Mathematics,* 1**,** 187-194.

MONAGHAN, J. J. & RAFIEE, A. 2013. A simple SPH algorithm for multi-fluid flow with high density ratios. *International Journal for Numerical Methods in Fluids,* 71**,** 537-561.

MORRIS, J. P. 1996. A study of the stability properties of smooth particle hydrodynamics. *Publications Astronomical Society of Australia,* 13**,** 97-102.

MORRIS, J. P. 2000. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids,* 33**,** 333-353.

MORRIS, J. P., FOX, P. J. & ZHU, Y. 1997. Modeling low Reynolds number incompressible flows using SPH. *Journal of Computational Physics,* 136**,** 214-226.

MUHAMMAD, N., ROGERS, B. D. & LI, L. 2013. Understanding the behaviour of pulsed laser dry and wet micromachining processes by multi-phase smoothed particle hydrodynamics (SPH) modelling. *Journal of Physics D-Applied Physics,* 46.

NAYROLES, B., TOUZOT, G. & VILLON, P. 1991. THE DIFFUSE ELEMENTS METHOD. *Comptes Rendus De L Academie Des Sciences Serie Ii,* 313**,** 133-138.

NGUYEN, V. P., RABCZUK, T., BORDAS, S. & DUFLOT, M. 2008. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation,* 79**,** 763-813.

NICKLIN, D. J. 1962. 2-PHASE BUBBLE FLOW. *Chemical Engineering Science,* 17**,** 693-702.

NUGENT, S. & POSCH, H. A. 2000. Liquid drops and surface tension with smoothed particle applied mechanics. *Physical Review E,* 62**,** 4968-4975.

NVIDIA 2012. CUDA C Programming Guide.

OGER, G., DORING, M., ALESSANDRINI, B. & FERRANT, P. 2007. An improved SPH method: Towards higher order convergence. *Journal of Computational Physics,* 225**,** 1472-1492.

OGER, G., JACQUIN, E., DORING, M., GUILCHER, P. M., DOLBEAU, R., CABELGUEN, P. L., BERTAUX, L., LE TOUZE, D. & ALESSANDRINI, B. 2010. Parallel hybrid CPU/GPU acceleration of the 3-D parallel code SPH-flow. *5th international SPHERIC Workshop.* Manchester, UK.

OMIDVAR, P. 2010. *Wave loading on bodies in the free surface using smoothed particle hydrodynamics.* Doctor of Philosophy, University of Manchester.

ONATE, E., IDELSOHN, S., ZIENKIEWICZ, O. C. & TAYLOR, R. L. 1996a. A finite point method in computational mechanics. Applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering,* 39**,** 3839-3866.

ONATE, E., IDELSOHN, S., ZIENKIEWICZ, O. C., TAYLOR, R. L. & SACCO, C. 1996b. A stabilized finite point method for analysis of fluid mechanics problems. *Computer Methods in Applied Mechanics and Engineering,* 139**,** 315-346.

PAULING, J. 2008. *Strength of ships and ocean structures,* Jersey City, US, New Jersey Society of Naval Architects and Marine Engineers.

PEREGRINE, D. H. 1967. LONG WAVES ON A BEACH. *Journal of Fluid Mechanics,* 27**,** 815-&.

PEREGRINE, D. H. 2003. Water-wave impact on walls. *Annual Review of Fluid Mechanics,* 35**,** 23-43.

PEREGRINE, D. H. & COOKER, M. J. 1990. Violent Water Motion at Breaking-Wave Impact. *22nd International Conference on coastal engineering.*

PEREGRINE, D. H. & THAIS, L. 1996. The effect of entrained air in violent water wave impacts. *Journal of Fluid Mechanics,* 325**,** 377-397.

PERLMAN, M. 1985. ON THE ACCURACY OF VORTEX METHODS. *Journal of Computational Physics,* 59**,** 200-223.

PLOUMHANS, P. & WINCKELMANS, G. S. 2000. Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal of Computational Physics,* 165**,** 354-406.

PRICE, D. J. & MONAGHAN, J. J. 2004. Smoothed particle magnetohydrodynamics - II. Variational principles and variable smoothing-length terms. *Monthly Notices of the Royal Astronomical Society,* 348**,** 139-152.

QIAN, L. F., BATRA, R. C. & CHEN, L. M. 2004. Static and dynamic deformations of thick functionally graded elastic plates by using higher-order shear and normal deformable plate theory and meshless local Petrov-Galerkin method. *Composites Part B-Engineering,* 35**,** 685-697.

RABCZUK, T. & EIBL, J. 2003. Simulation of high velocity concrete fragmentation using SPH/MLSPH. *International Journal for Numerical Methods in Engineering,* 56**,** 1421-1444.

RANDLES, P. W. & LIBERSKY, L. D. 1996. Smoothed particle hydrodynamics: Some recent improvements and applications. *Computer Methods in Applied Mechanics and Engineering,* 139**,** 375-408.

RANDLES, P. W. & LIBERSKY, L. D. 2000. Normalized SPH with stress points. *International Journal for Numerical Methods in Engineering,* 48**,** 1445-+.

RASIO, F. A. & SHAPIRO, S. L. 1991. COLLISIONS OF GIANT STARS WITH COMPACT OBJECTS - HYDRODYNAMICAL CALCULATIONS. *Astrophysical Journal,* 377**,** 559-580.

REYHANOGLU, M. 2003. Maneuvering control problems for a spacecraft with unactuated fuel slosh dynamics. *Cca 2003: Proceedings of 2003 Ieee Conference on Control Applications, Vols 1 and 2***,** 695-699.

RITTER, A. 1892. Die fortpflanzung der wasserwellen. *Zeitschrift des Vereins Deutscher Ingenieure,* 36**,** 947-954.

ROGERS, B. D., LEDUC, J., MARONGIU, J. C. & LEBOEUF, F. 2009. Comparison and evaluation of multi-phase and surface tension models. *4th SPHERIC Workshop.* Nantes, France.

ROONEY, E. A., WEI, W. Y., IRISH, J. L., WEISS, R., DALRYMPLE, R. A., HERAULT, A. & BILOTTA, G. 2011. Testing accuracy and convergence of GPUSPH for free-surface flows.

ROSENHEAD, L. 1931. The formation of vortices from a surface of discontinuity. *Proceedings of the Royal Society of London Series a-Containing Papers of a Mathematical and Physical Character,* 134**,** 170-192.

ROSSINELLI, D., BERGDORF, M., COTTET, G.-H. & KOUMOUTSAKOS, P. 2010. GPU accelerated simulations of bluff body flows using vortex particle methods. *Journal of Computational Physics,* 229**,** 3316-3333.

SCHOENBERG, I. J. 1946. CONTRIBUTIONS TO THE PROBLEM OF APPROXIMATION OF EQUIDISTANT DATA BY ANALYTIC FUNCTIONS .A. ON THE PROBLEM OF SMOOTHING OR GRADUATION - A 1ST CLASS OF ANALYTIC APPROXIMATION FORMULAE. *Quarterly of Applied Mathematics,* 4**,** 45-99.

SHADLOO, M. S., ZAINALI, A., SADEK, S. H. & YILDIZ, M. 2011. Improved Incompressible Smoothed Particle Hydrodynamics method for simulating flow around bluff bodies. *Computer Methods in Applied Mechanics and Engineering,* 200**,** 1008-1020.

SHADLOO, M. S., ZAINALI, A., YILDIZ, M. & SULEMAN, A. 2012. A robust weakly compressible SPH method and its comparison with an incompressible SPH. *International Journal for Numerical Methods in Engineering,* 89**,** 939-956.

SHAO, S. 2012. Incompressible smoothed particle hydrodynamics simulation of multifluid flows. *International Journal for Numerical Methods in Fluids,* 69**,** 1715-1735.

SHAO, S. D. & LO, E. Y. M. 2003. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources,* 26**,** 787-800.

SHEPARD, D. A two dimensional interpolation function for irregularly spaced data. 23rd Nat. Conf. ACM, 1968. 517-523.

SIMS, J. S. & MARTYS, N. 2004. Simulation of sheared suspensions with a parallel implementation of QDPD. *Journal of Research of the National Institute of Standards and Technology,* 109**,** 267-277.

SKILLEN, A., LIND, S., STANSBY, P. K. & ROGERS, B. D. 2013. incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothingapplied to body-water slam and efficient wave-body interaction. *Computer methods in applied mehanics and engineering*.

SMITH, P. A. & STANSBY, P. K. 1988. IMPULSIVELY STARTED FLOW AROUND A CIRCULAR-CYLINDER BY THE VORTEX METHOD. *Journal of Fluid Mechanics,* 194**,** 45-77.

SOUTO-IGLESIAS, A., BOTIA-VERA, E. & BULIAN, G. 2011a. Repeatability and two dimensionality of model scale sloshing impacts. *International offshore and polar engineering conference.* The international society of offshore and polar engineers.

SOUTO-IGLESIAS, A., BOTIA-VERA, E., MARTIN, A. & PEREZ-ARRIBAS, F. 2011b. A set of canonical problems in sloshing. Part 0: Experimental setup and data processing. *Ocean Engineering,* 38**,** 1823-1830.

SOUTO-IGLESIAS, A., MACIA, F., GONZALEZ, L. M. & CERCOS-PITA, J. L. 2013. On the consistency of MPS. *Computer Physics Communications,* 184**,** 732-745.

SPEZIALE, C. G. 1987. ON THE ADVANTAGES OF THE VORTICITY VELOCITY FORMULATION OF THE EQUATIONS OF FLUID-DYNAMICS. *Journal of Computational Physics,* 73**,** 476-480.

SPRINGEL, V. 2005. The cosmological simulation code GADGET-2. *Monthly Notices of the Royal Astronomical Society,* 364**,** 1105-1134.

STANSBY, P. K., CHEGINI, A. & BARNES, T. C. D. 1998. The initial stages of dam-break flow. *Journal of Fluid Mechanics,* 374**,** 407-424.

SUESS, M. & LEOPOLD, C. 2008. Common mistakes in OpenMP and how to avoid them. *Openmp Shared Memory Parallel Programming, Proceedings,* 4315**,** 312-323.

SUSSMAN, M., SMEREKA, P. & OSHER, S. 1994. A LEVEL SET APPROACH FOR COMPUTING SOLUTIONS TO INCOMPRESSIBLE 2-PHASE FLOW. *Journal of Computational Physics,* 114**,** 146-159.

SWEGLE, J. W., HICKS, D. L. & ATTAWAY, S. W. 1995. SMOOTHED PARTICLE HYDRODYNAMICS STABILITY ANALYSIS. *Journal of Computational Physics,* 116**,** 123-134.

TAKEDA, H., MIYAMA, S. M. & SEKIYA, M. 1994. NUMERICAL-SIMULATION OF VISCOUS-FLOW BY SMOOTHED PARTICLE HYDRODYNAMICS. *Progress of Theoretical Physics,* 92**,** 939-960.

TARTAKOVSKY, A. M., FERRIS, K. F. & MEAKIN, P. 2009. Lagrangian particle model for multiphase flows. *Computer Physics Communications,* 180**,** 1874-1881.

TITOV, V. V. & SYNOLAKIS, C. E. 1998. Numerical modeling of tidal wave runup. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 124**,** 157-171.

TIWARI, S. & KUHNERT, J. 2007. Modeling of two-phase flows with surface tension by finite pointset method (FPM). *Journal of Computational and Applied Mathematics,* 203**,** 376-386.

VACONDIO, R. & MIGNOSA, P. 2009. Finite pointset method for free surface flow. *4th International SPHERIC Workshop.* Nantes, France.

VACONDIO, R., ROGERS, B. D. & STANSBY, P. K. 2012. Smoothed Particle Hydrodynamics: Approximate zero-consistent 2-D boundary conditions and still shallow-water tests. *International Journal for Numerical Methods in Fluids,* 69**,** 226-253.

VACONDIO, R., ROGERS, B. D., STANSBY, P. K., MIGNOSA, P. & FELDMAN, J. 2013. Variable resolution for SPH: A dynamic particle coalescing and splitting scheme. *Computer Methods in Applied Mechanics and Engineering,* 256**,** 132-148.

VAJDA, A. 2011. *Programming many-core chips,* Dordrecht, Springer.

VALDEZ-BALDERAS, D., DOMINGUEZ, J. M., CRESPO, A. J. C. & ROGERS, B. D. 2011. Developing massively parallel SPH simulations on multi-GPU clusters. *6th ERCOFTAC SPHERIC Workshop.* Hamburg, Germany.

VERLET, L. 1967. COMPUTER EXPERIMENTS ON CLASSICAL FLUIDS .I. THERMODYNAMICAL PROPERTIES OF LENNARD-JONES MOLECULES. *Physical Review,* 159**,** 98-&.

VICCIONE, G., BOVOLIN, V. & CARRATELLI, E. P. 2008. Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations. *International Journal for Numerical Methods in Fluids,* 58**,** 625-638.

VILA, J. P. 1999. On particle weighted methods and smooth particle hydrodynamics. *Mathematical Models & Methods in Applied Sciences,* 9**,** 161-209.

VIOLEAU, D. & ISSA, R. 2007a. Numerical modelling of complex turbulent free-surface flows with the SPH method: an overview. *International Journal for Numerical Methods in Fluids,* 53**,** 277-304.

VIOLEAU, D. & ISSA, R. 2007b. SPHERIC benchmark test case number 5: sensistivity analysis to numerical and physical parameters. *2nd Interational SPHERIC Workshop.* Madrid, Spain.

VON NEUMANN, J. & RICHTMYER, R. D. 1950. A METHOD FOR THE NUMERICAL CALCULATION OF HYDRODYNAMIC SHOCKS. *Journal of Applied Physics,* 21**,** 232-243.

WEMMENHOVE, R. 2008. *Numerical prediction of two-phase flow in offshore environments.* Rijksuniversiteit Groningen.

WENDLAND, H. 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics,* 4**,** 389-396.

WIEGEL, R. L. 1992. *Oceanographical Engineering*, Dover Publications.

WIRSCHING, P. H. 1984. FATIGUE RELIABILITY FOR OFFSHORE STRUCTURES. *Journal of Structural Engineering-Asce,* 110**,** 2340-2356.

WOOD, D. J., PEREGRINE, D. H. & BRUCE, T. 2000. Wave impact on a wall using pressure-impulse theory. I: Trapped air. *Journal of Waterway Port Coastal and Ocean Engineering-Asce,* 126**,** 182-190.

XU, R., STANSBY, P. & LAURENCE, D. 2009. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics,* 228**,** 6703-6725.

ZHANG, S., MORITA, K., FUKUDA, K. & SHIRAKAWA, N. 2006. An improved MPS method for numerical simulations of convective heat transfer problems. *International Journal for Numerical Methods in Fluids,* 51**,** 31-47.