



Imperial College London
Department of Aeronautics
November 2016

Sloshing simulations with the smoothed particle hydrodynamics (SPH) method

Mashy David Green

This thesis is submitted for the degree of
Doctor of Philosophy of Imperial College London.

Dedicated to Inbar and Oliver Green. Their smiles remind me to
always keep doing what I love and never give up on my dreams.

Declaration

I hereby declare that all the work in this thesis is my own. All other sources of information used in this work have been fully acknowledged.

The copy right of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Mashy D Green

Acknowledgements

First and foremost I would like to thank my supervisor, Dr Joaquim Peiró, for providing me with both academic and personal support far beyond my expectations.

I would also like to thank the following people. Prof. Javier Bonet of Greenwich University for the perusal of his SPHere code, which gave me the first glimpse into the wonderful world of SPH. Dr Stefan Adami and Dr Xiangyu Hu, both of Technische Universität München, for their help with the understanding and implementation of their SPH formulation. Jan Eichstädt for developing a Matlab code as part of his MSc project, that has allowed me to identify and solve numerical issues, and test new formulations, before committing them to the GPU code.

I am hugely grateful to Dr José Manuel Domínguez Alonso of Universidad de Vigo for all the technical assistance, and for providing me with a pre-release double precision version of DualSPHysics (version 4). I am thankful to Dr Benedict Rogers, Dr Alejandro Crespo, and the DualSPHysics team for the opportunity to collaborate with them and for welcoming me with warm hospitality in Manchester and Ourense. It was truly a pleasure working with this group of people.

I would also like to thank the ESA/ESTEC team: José Longo, Chiara Lada and Andrea Passaro, both for sponsoring this project and for their support and their contributions to our technical discussions.

Lastly, I would like to thank my family, and especially my wife and my parents, for the infinite amount of support at difficult times, providing me with motivation and inspiration when I needed it the most. I could not have done it without you.

Abstract

The main aims of this work are to identify, verify, and validate a smoothed particle hydrodynamics (SPH) method for simulating long duration transient and steady-state fluid sloshing in complex geometries. The validation will be carried out by comparing the SPH simulations against experimental data provided by ESA/ESTEC for transient and steady-state sloshing in a rectangular tank with a low filling ratio and of transient sloshing in a pill-shaped tank that exhibits transition from swaying to swirling waves.

The experimental tests proved to be extremely challenging due to the low fill ratio of the rectangular tank and the long duration of both experiments. The main challenge is to devise a SPH scheme that balances spatial and temporal accuracy with an efficient computer implementation to produce accurate simulations at a reasonable computing cost.

The investigation highlighted three issues of critical importance: the treatment of solid boundaries in order to limit the introduction numerical errors into the system; the application of a correct numerical dissipation scheme to reduce existing numerical errors; and the need for a massively parallel implementation..

Careful examination of the most suitable techniques led to the adoption a corrected δ -SPH scheme that provides numerical dissipation to reduce spurious pressure oscillations, and a fixed ghost particle boundary condition to accurately impose wall boundary conditions. The proposed SPH methods were coded in the open source parallel code DualSPHysics.

The implementation showed significant improvements in energy conservation and solution accuracy when compared to state-of-the-art SPH methods, and accurately reproduced known analytical solutions to linear sloshing.

The validation against the ESA/ESTEC experimental data showed excellent agreement between the SPH simulations and experiments, accurately reproducing the time history of wave heights and sloshing forces as well as capturing the full free-surface shapes. Only the careful selection of appropriate boundary conditions, artificial dissipation and a massively parallel GPU architecture allowed to accurately simulate these experiments.

Contents

1	Introduction, objectives and scope	1
2	Literature Review	4
2.1	Sloshing	4
2.2	CFD for simulating sloshing	6
2.3	The Smoothed Particle Hydrodynamics (SPH) method	7
2.3.1	SPH for sloshing problems	9
2.3.2	Massively parallel SPH	10
3	Smoothed particle hydrodynamics (SPH)	12
3.1	Basic SPH formulation	12
3.1.1	Governing equations of weakly compressible flow	12
3.1.2	SPH discretisation	13
3.1.3	Governing equations in semi-discrete form	16
3.1.4	Smoothing kernel function and its derivative	16
3.1.5	General formulation of the gradient and divergence operators .	19
3.1.6	Governing equations in WCSPH formalism	20
3.1.7	Lagrangian derivation of the SPH operators	22
3.2	Spatial numerical errors in WCSPH	24
3.2.1	Discretisation errors	24
3.2.2	Consistency errors	25
3.3	Free-surface boundary conditions in WCSPH	26
3.4	Wall boundary conditions	27
3.4.1	Wall boundary conditions review	28
3.4.2	Fixed ghost particle method with force balance correction . .	33
3.4.3	Fixed ghost particle boundary condition errors	35
3.5	Dissipation models	36

3.5.1	Artificial viscosity	37
3.5.2	Laminar Viscosity	37
3.5.3	Density dissipation	38
3.5.4	Damped hydrostatic initialisation	41
3.6	Governing equations in a moving frame of reference	41
3.7	Time integration	42
3.8	Post-processing	44
3.8.1	Free-surface detection and height measurements	44
3.8.2	Energy calculation	47
3.8.3	Force calculations	49
4	Parallel implementation on GPUs	52
4.1	GPU architecture and programming	53
4.2	SPH on GPUs	56
4.3	DualSPHysics open-source code	58
4.4	DualSPHysicsGPM code	62
4.4.1	GPBC implementation	62
4.4.2	Corrected δ -SPH implementation	63
4.5	Single versus double precision	67
4.6	Performance of DualSPHysicsGPM	69
5	Verification and validation	72
5.1	Hydrostatic simulations	73
5.1.1	Hydrostatic boundary condition verification	73
5.1.2	Hydrostatic boundary condition comparison	74
5.1.3	Hydrostatic δ -SPH comparison	76
5.2	Linear sloshing	84
5.3	Dam break	86
5.4	3D dam break - SPHERIC Test 2	91
6	Validation against sloshing experiments	97
6.1	Rectangular tank long duration	99
6.2	Rectangular tank three-dimensional steady-state simulations	108
6.3	Sloshing in a pill-shaped tank	112
7	Conclusions	121
7.1	Future work	123

List of Figures

3.1	The particle-based SPH approximation and notation.	15
3.2	The two-dimensional Wendland kernel and its derivative	18
3.3	Kernel function's compact support in the vicinity of a free surface. . .	26
3.4	Kernel function's compact support in the proximity of a solid boundary.	28
3.5	Illustration of different wall boundary condition categories	29
3.6	Illustration of the fixed ghost particles boundary condition [3].	34
3.7	A pressure interpolation post processing example.	45
3.8	The free surface of a dam break.	46
3.9	Example of the total energy over time for still water cases comparing different boundary conditions.	49
3.10	Comparison of different methods to compute fluid forces in SPH. . . .	51
4.1	CPU vs GPU transistor allocation.	53
4.2	GPU threads blocks and memory.	55
4.3	Flow diagram of the GPU implementation.	59
4.4	Call function graph for for DualSPHysics (version 4.0).	60
4.5	Call function graph for DualSPHysicsGPM.	65
4.6	Single precision errors in the pressure field for different positions of the origin.	68
5.1	Pressure profiles at centre of fluid domain during a warm-up stage in a hydrostatic simulation using a fixed ghost particle boundary condition.	74
5.2	Pressure field at a hundred seconds of hydrostatic simulations using different boundary conditions.	77
5.3	Pressure profiles at a hundred of hydrostatic simulations using differ- ent boundary conditions.	77
5.4	Energy components over time of hydrostatic simulations using differ- ent boundary conditions.	78

5.5	Pressure field at a hundred of hydrostatic simulations using different numerical dissipation schemes.	81
5.6	Pressure profiles at a hundred of hydrostatic simulations using different numerical dissipation schemes.	81
5.7	Total energy over time of hydrostatic simulations using different numerical dissipation schemes.	82
5.8	Energy components over time in the first ten seconds of hydrostatic simulations using different numerical dissipation schemes.	82
5.9	Energy components over time of hydrostatic simulations using different numerical dissipation schemes.	83
5.10	Linear sloshing wave elevation over time for the SPH simulation and the analytical solution.	85
5.11	Linear sloshing fluid force over time.	85
5.12	Accumulated error of fluid forces in a linear sloshing simulation convergence study.	86
5.13	Schematic of the dimensions and initialisation of the dam break simulation.	87
5.14	Snapshots of the pressure field in a dam break simulation using the fixed ghost particle boundary condition.	89
5.15	Pressure over time for a dam break simulation against experimental data using the fixed ghost particles boundary condition.	90
5.16	Pressure over time for a dam break simulation against experimental data using the corrected Antuono δ -SPH scheme.	90
5.17	Sketch of the three-dimensional dam break SPHERIC validation test case - SPHERIC Test 2.	93
5.18	Comparison of simulated and experimental water heights over time for SPHERIC Test 2.	94
5.19	Comparison of simulated and experimental pressure over time for SPHERIC Test 2.	95
5.20	Two-dimensional slice of the SPH simulations for SPHERIC Test 2 showing the interpolated heights.	96
5.21	Two-dimensional slice of the SPH simulations for SPHERIC Test 2 showing the interpolated pressures.	96
6.1	Schematics of the rectangular and pill-shaped tank used for sloshing experiments.	98

6.2	Rectangular tank normalised displacement for the frequencies $f_w = 0.32$, $f_w = 0.46\text{Hz}$ and $f_w = 0.55\text{Hz}$	99
6.3	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$ at location H1.	102
6.4	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$ at location H2.	102
6.5	Sloshing force over time of the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$	103
6.6	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$ at location H1.	103
6.7	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$ at location H2.	104
6.8	Sloshing force over time of the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$	104
6.9	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$ at location H1.	105
6.10	Water heights over time in the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$ at location H2.	105
6.11	Sloshing force over time of the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$	106
6.12	Snapshots of the SPH simulation with excitation frequency $f_{w2} = 0.46\text{Hz}$ showing height measurements at location H1: (left) taken at time $t = 293.1\text{s}$; (right) taken at $t = 295.3\text{s}$	106
6.13	Snapshots of the rectangular tank experiment for a whole cycle after reaching a steady state under an excitation frequency of $f_{w2} = 0.46\text{Hz}$ with the SPH free-surface overlaid	107
6.14	Snapshots of the rectangular tank experiment for a whole cycle after reaching a steady state under an excitation frequency of $f_{w3} = 0.55\text{Hz}$ with the SPH free-surface overlaid	107
6.15	Water heights over time in the rectangular tank sloshing experiment with an excitation frequency $f_{w2} = 0.46\text{Hz}$. Steady state comparison between the experiment and the three-dimensional simulation.	109
6.16	Sloshing forces over time in the rectangular tank sloshing experiment with an excitation frequency $f_{w2} = 0.46\text{Hz}$. Steady state comparison between the experiment and the three-dimensional SPH simulation.	109

6.17	Snapshots of the density field from the three-dimension SPH simulation of the rectangular tank sloshing experiment under an excitation frequency of $f_{w2} = 0.46\text{Hz}$. Comparison between artificial viscosity and the corrected δ -SPH scheme.	110
6.18	Sloshing force over time in the rectangular tank sloshing experiment under an excitation frequency of $f_{w2} = 0.46\text{Hz}$. Steady state comparison between artificial viscosity and the corrected δ -SPH scheme. . . .	110
6.19	Water heights over time in the rectangular tank sloshing experiment under an excitation frequency of $f_{w2} = 0.46\text{Hz}$. Steady state comparison between artificial viscosity and the corrected δ -SPH scheme. . . .	111
6.20	Time history of the excitation frequency over the full duration for the pill-shaped tank sloshing experiment.	113
6.21	Time history of the peak displacement and excitation frequency for the pill-shaped tank sloshing experiment over the duration of the simulation.	113
6.22	Sloshing force components over time for the pill-shaped tank sloshing experiment simulation showing the full one hundred seconds duration.	114
6.23	Sloshing force components over time for the pill-shaped tank sloshing experiment simulation, showing the time interval $10 \leq t \leq 35$ seconds.	115
6.24	Sloshing force components over time for the pill-shaped tank sloshing experiment simulation, showing the time interval $35 \leq t \leq 50$ seconds.	116
6.25	Side-by-side snapshots from the pill-shaped tank sloshing experiment alongside the SPH simulation. Taken around the time the breaking waves trigger the transition into swirling.	117
6.26	Sloshing force components over time for the pill-shaped tank sloshing experiment simulation, showing the time interval $55 \leq t \leq 75$ seconds.	118
6.27	Sloshing force components over time for the pill-shaped tank sloshing experiment simulation, showing the time interval $80 \leq t \leq 100$ seconds.	119
6.28	The frequency domain of the x -axis sloshing force component for the pill-shaped tank sloshing experiment.	119
6.29	The frequency domain of the y -axis sloshing force component for the pill-shaped tank sloshing experiment.	120
6.30	The frequency domain of the z -axis sloshing force component for the pill-shaped tank sloshing experiment.	120

List of Tables

3.1	Comparison of solid walls boundary condition features.	33
4.1	Simplified overview of the main functions called in DualSPHysics (version 4.0) GPU code using the symplectic time-stepping scheme. . . .	61
4.2	Simplified overview of the main functions called in DualSPHysicsGPM code using the symplectic time-stepping scheme and corrected δ -SPH. . . .	66
4.3	Comparison of performance for different features of the code in large 2D and 3D cases.	70
4.4	Comparison of performance for different features of the code in a small 2D case on a Tian Z GPU.	71
4.5	Comparison of performance for different features of the code in a small 2D case on a GTX 970 GPU.	71
6.1	Specifications of the rectangular tank geometry.	99
6.2	Specifications of the pill-shaped tank geometry.	99

Chapter 1

Introduction, objectives and scope

Liquid sloshing occurs in almost all forms of land, air and sea fuel powered transportation, satellites and space vehicles, rockets and missiles, land and sea based liquid containing structures during earthquakes, hurricanes, and landslides, and also in lakes, harbours and more [43]. The motivation for this work is the advancement of computational fluid dynamics tools capable of simulating sloshing, in particular for satellites and space vehicles in micro-gravity conditions. Micro-gravity sloshing of cryogenic fuels can cause dynamic instabilities and lose of fluid, and has been identified as a critical component for improving the safety and efficiency of future satellites and re-ignitable upper stages [106].

Currently, no single method is capable of dealing with the full range of physics problems required for the study of cryogenic micro-gravity sloshing. These include long duration transient sloshing in real fuel tank geometries (three dimensional), multi-fluids (liquid and gas phases), surface tension dominated flows and the effects of the triple point contact angle (solid, fluid, gas), heat transfer and its effects on surface tension and the triple point, and lastly vaporisation and phase change. Each of these problems requires simulating different physical phenomena which involve a wide range of time and space scales, making this a complex multi-physics multi-scale problem.

Smoothed particles hydrodynamics (SPH) is a mesh-free Lagrangian method which has been identified as having potential for simulating cryogenic micro-gravity sloshing problems. As a starting point, the capabilities of the method for different Earth-gravity sloshing cases with a single fluid should be fully explored and understood to provide a solid foundation before advancing the method to include the full range of requirements. The main objective of this thesis is to investigate the suit-

ability of the state-of-the-art in SPH for simulating these problems. While there is a substantial body of research into Earth-gravity sloshing with SPH, it is generally limited in scope focusing on short duration or steady-state sloshing in rectangular tanks with high fill ratios. A review of some of the work done in this field is found in Section 2.3.1.

The specific objective of this work is to expand the study of sloshing with SPH to include: sloshing in tanks with low fill ratios; long duration transient and steady-state sloshing in rectangular tanks; sloshing in three dimensional pill-shaped fuel tanks; the transition from planar to swirling motion; and long duration transient sloshing with swirling.

A careful examination of the limitations of the SPH method for achieving the above objectives is required, and the best available solutions proposed in the literature are to be identified. The identified features should be implemented into an all inclusive SPH scheme capable of meeting the desired goals.

All these objectives require long computational times and thus need to be implemented in efficient parallel codes. This is the reason to choose the open source SPH solver suite DualSPHysics [28] as the development platform for this work.

DualSPHysics is a hybrid parallel code that runs on central processing units (CPU) or graphical processing units (GPU), and is written in C++ with OpenMP [10] and CUDA [109]. It is considered the state-of-the-art parallel SPH open-source code. The identified requirements and features of the SPH method are to be included into this code, advancing its capabilities with the ultimate goal of developing and validating a SPH method suitable for modelling sloshing.

A modified version of DualSPHysics, which will be referred to in this thesis as DualSPHysicsGPM, is developed. It includes an improved boundary condition treatment with fixed ghost particles and a force balance correction [3], and a corrected numerical dissipation scheme [8] to reduce spurious pressure oscillations. This code is capable of accurately simulating challenging sloshing problems and is validated against experiments by ESA/ESTEC of long duration sloshing that exhibit a range of complex flow phenomena.

The layout of the thesis is the following. Chapter 2 is a literature review covering the sloshing phenomena, numerical methods for simulating sloshing and the SPH method. Chapter 3 provides a description and the derivation of the SPH method, discussing its most important features and limitations. Two of the main issues with the SPH method are identified and discussed in further detail: wall boundary con-

ditions and numerical dissipation methods. The most suitable solutions found in the literature for these problems are highlighted and described. A discussion of the post-processing procedure for SPH simulation data concludes this chapter.

Chapter 4 provides a general overview of GPU architecture and programming practices required for the efficient implementation of SPH algorithms. This is followed by a brief description of organisation of the DualSPHysics code, and its modified version DualSPHysicsGPM, which includes the additional solid boundary condition and numerical dissipation scheme identified in Chapter 3. A discussion on numerical precision and its strict requirement for the cases of interest is then given, followed by performance benchmarks of the DualSPHysicsGPM code against the original.

Chapter 5 presents a series of verification and validation tests using DualSPHysicsGPM. The tests cover comparison against analytical solutions and experimental data. Where possible the results are also benchmarked against the unmodified DualSPHysics code, showing improvements of the key features.

Chapter 6 describes the validation of the code against experimental data for a rectangular and pill-shape tanks subjected to sinusoidal lateral excitation in a range of amplitudes and frequencies that covers the stated objectives.

Finally, conclusions and suggestions for further work are given in Chapter 7.

Related published work

M.D. Green and J. Peiro. Fixed ghost particles boundary conditions in DualSPHysics. In *the proceedings of the First DualSPHysics Users Workshop*, University of Manchester, 19 – 20 January, 2015.

M.D. Green, J.M. Dominguez, A.J. Crespo and J. Peiro. Long duration SPH simulations of sloshing in tanks with a low fill ratio and high stretching. In *the proceedings of the eleventh International SPHERIC Workshop*, pages 175 – 182, Technical University of Munich, Germany, 14 – 16 June, 2016.

M.D. Green and J. Peiro. Three-dimensional simulation of slosh dynamics using a modified DualSPHysics code. In *the proceedings of the Second DualSPHysics Users Workshop*, University of Manchester, 6 – 7 December, 2016.

M.D. Green and J. Peiro. Long duration SPH simulations of sloshing in tanks with a low fill ratio and high stretching. In preparation for submission to *Journal of Computational Physics*.

Chapter 2

Literature Review

This literature review discusses the sloshing phenomena, its modelling and numerical simulation, and gives a brief overview of the research conducted in the area. A brief review of computational fluid dynamics (CFD) methods for simulating sloshing is then given, highlighting the features required for these types of simulations. Lastly, the SPH method is introduced, discussing the main advantages and drawbacks of the method, reviewing the use of SPH for sloshing simulations, and the advancements made in the parallel implementation of the method.

2.1 Sloshing

The term sloshing refers to free-surface flows in partially filled containers undergoing motions generated by external forcing. Most of the work on sloshing is concerned with large liquid-filled structures such as rocket fuel tanks, which are subject to considerable accelerations and vibrations. Of particular interest is the prediction of forces and torques due to liquids experiencing various sloshing motions and the suppression of sloshing necessary for vehicle control and fuel management.

Sloshing has been of interest to researchers for the last 50 years [53] and has recently become a very active research area in many mechanical engineering disciplines including aerospace, marine and civil [43]. The effects of liquid sloshing are the generation of dynamic loads on the containing structure [135] and violent behaviour of the fluid's free surface. It can cause structural failure, flight and manoeuvrability instabilities, ship capsizing, fluid and gas mixing and a verity of other problems. For example, the near disaster of the NEAR (Near Earth Asteroid Rendezvous) mission to the asteroid Eros in 1998 was attributed to sloshing motion, which resulted in a

13 months delay [9].

The strongest effects of sloshing occur when the forcing frequency is close to the natural sloshing modes. This problem can usually be prevented by designing the liquid-containing structure and filling level to have natural sloshing modes which are extremely unlikely to be excited by the predicted phenomena (i.e. wind, waves, earthquakes etc.). However, in many cases this design approach is not sufficient, for instance when the filling level of the fluid changes during the operational time-scale, thus drastically affecting the natural modes. The appearance of swirling (or rotary) waves may occur during harmonic horizontal excitation in vertically axisymmetric tanks such as upright cylinders and spheres [1]. The swirling waves are triggered by the breakdown in symmetry and non-linearity of the fluid motion when the excitation is in the vicinity of the lowest natural sloshing mode. These waves generate a hydrodynamic force perpendicular to the direction of the forced excitation. The simulation of this behaviour is one of the key objectives of this thesis.

The motion of a fluid inside a container is governed by the Navier-Stokes equations together with appropriate solid walls and free-surface boundary conditions. Analytical solutions based on the potential flow theory and multimodal methods exist, but are only applicable in quite restrictive and simple cases where only small and relatively smooth sloshing waves are generated [43].

Early research was motivated by fuel sloshing in aircraft and rockets [1], and later focused more on marine applications such as in liquefied natural gas carriers [113] and tuned liquid dampers [121]. More recently, sloshing has been investigated in a much wider context ranging from fuel storage bays in nuclear reactors subjected to earthquakes [42], automotive fuel cell sloshing [85] and microgravity sloshing in satellites and re-ignitable upper stages [76]. Extensive experimental research was conducted covering a wide range of phenomena such as slamming, air cavities and three-dimensional effects from the walls [134]. Experiments in different geometries such as cylindrical, spherical and pill-shaped tanks can be found in references [1, 113, 128, 129]. Liquid sloshing experiments in microgravity have been pioneered by both NASA and the European Space Agency (ESA) in the Sloshsat FLEVO mission [41] and more recently in the SPHERES experiment aboard the international space station [131]. Ground based experiments of sloshing in full-sized fuel tanks have also been conducted by ESA/ESTEC [77] which are used for validation in this thesis. While our understanding of sloshing is not complete, it is mature enough to be the subject of textbooks [43, 68] and is continuously expanding. The ability of producing

accurate simulations of sloshing will certainly help to further this understanding.

2.2 CFD for simulating sloshing

A very extensive and well-organized review of CFD methods for the simulation of liquid sloshing can be found in Chapter 10 of reference [43] that covers both grid-based and particle-based (or meshless) methods. A more succinct review of numerical simulation of sloshing in partially filled liquid containers which also includes SPH methods can be consulted in [120]. Here only a brief overview is given.

There are mainly two types of formulations for numerically modelling free surfaces and moving interfaces: Lagrangian and Eulerian. Lagrangian techniques move the grid points to follow the fluid and accurately represent the shape of the interface. For large motions of the interface, the grid could be subject to severe distortion that might lead to numerical inaccuracies and instabilities. In the case of Eulerian techniques, the mesh is fixed and the interface travels across the mesh. However, the interface must be tracked as its position is required to know where to apply the balance laws and boundary conditions. Mixed methods such the arbitrary Lagrangian-Eulerian technique that attempt to get the best qualities of both techniques have also been proposed [15].

However, the limitations of the Lagrangian approach in this context means that Eulerian techniques have been favoured, following the pioneering work in the Los Alamos National Laboratory started in the late eighties, for the CFD simulation of multiphase flows involving a free surface or a fluid interface. A variety of such CFD methods are discussed in the reviews [55, 78, 130, 141, 143]. A comprehensive review is that of reference [149] which presents numerical methods and models for interface resolving simulations of multiphase flows in micro-fluidics.

Amongst the techniques developed to track the interfaces the following are highlighted. The marker-and-cell (MAC) method [59] where discrete markers were used to identify the different fluids in the grid cells. The volume of fluid (VOF) method [63] that uses a piecewise constant approximation for the free surface. The VOF approach has been later improved by introducing a piecewise linear interface calculation (PLIC) scheme [124] and it is currently considered to be the state-of-the-art method. Reference [122] uses piecewise-parabolic segments to capture the interface. Alternative techniques for modelling the interface use a continuous representation of the interface. The level set method [137, 138] considers the interface to

be sharp and represents it as the zero level set of a smooth implicit function that propagates with a velocity perpendicular to its gradient. It can also be interpreted as a form of regularised Heaviside function. Interfaces can also be represented explicitly by the use of marker particles as in the front tracking method [142] where the interface is explicitly represented by a separate unstructured grid that moves through the fixed grid.

Grid-based CFD methods have reached a high degree of maturity and there are general-purpose commercial codes based on the algorithms described above, such as FLOW-3D [69], CFX and Fluent [6], STAR-CD [18], and FINE [108], and open-source codes, such as OpenFoam [25] and DROPS [55], to name but a few, that offer some of the capabilities required for the type of simulations of interest.

As an alternative to the grid-based methods mentioned above, various grid-less (or mesh-free) methods have been developed. These methods normally target applications where the grid-based methods no longer apply or require very complicated and computationally expensive techniques to solve. A good overview of different grid-less methods can be found in Chapter 1 of reference [81]. Amongst these methods, the smoothed particle hydrodynamics (SPH) is identified as being a strong candidate for modelling complex sloshing. The method is suitable for three-dimensional problems on a macroscopic scale. Its most attractive feature is the ability to deal with large deformations of the physical continuum without any special treatment, allowing it to naturally capture violent free-surface flows, liquid break-up and splashing. Its basic formulation is extremely easy to implement and is suitable for parallelisation, making it a very attractive choice. SPH also has a large research community [136] that actively improves and adds new capabilities to the method, and it is becoming increasingly popular for both academic and industrial applications.

2.3 The Smoothed Particle Hydrodynamics (SPH) method

SPH is a fully Lagrangian particle method for the solution of partial differential equations. Their discretisation uses Lagrangian integration points, called particles, which follow the fluid flow and the relevant properties are calculated at the location of each particle by a weighted average.

The origin of the SPH method can be traced back to references [50, 86], where SPH was conceived for modelling astrophysical problems and is still used for this purpose to this day [119]. It has since been adapted for modelling many other types of physical

and engineering problems such as free-surface and multiphase flows, and structural modelling. Three recent general reviews of the state-of-the-art SPH are [82], [101] and [147]. The first application of SPH to free-surface flows is described in [99] and the modelling of sloshing using SPH is reviewed in reference [52].

The discretisation into particles and the Lagrangian nature of the SPH method represent a physical system as a system of particles with all the associated properties. In this framework the principle of virtual displacement and the Lagrangian and Hamiltonian forms of the equations of motion, as described in every theoretical physics book such as [72], can be exploited to derive a variationally consistent formulations of the SPH scheme that will conserve mass, momentum, angular momentum and total energy. Such work has been carried out in the ground breaking work of Bonet et al. [12–14, 45, 75]. The SPH method also calculates advection exactly and also provides zero intrinsic dissipation and resolution that follows the mass [115]. All these features are clearly laid out in the SPH formulation derivation given in Section 3.1.

These seemingly desirable features are highly beneficial, however they can also lead to many undesirable side effects when dealing with closed domains and in the presence of a free surface. The introduction of solid walls and free-surface boundary conditions introduced numerical errors which, due to the exact advection property, can linger indefinitely in the fluid domain. The problem of boundary conditions in SPH is still open and is one of the grand challenges recognised by the SPH community [136]. The introduction of numerical errors, from the boundaries or otherwise, requires the addition of numerical dissipation to the SPH method. Generally, numerical dissipation is applied by an artificial diffusion term in either the momentum equation [98], the mass conservation equation or both [8]. However, many of the commonly used numerical dissipation schemes have limited applicability and violate some of the conservation properties that the method is required to preserve [7]. A discussion of the free-surface boundary condition is given in Section 3.3 and a review of some of the many solid wall boundary conditions in Section 3.4.1. A discussion of the most commonly used dissipation schemes and their properties is given in Section 3.5. Other numerical issues of the SPH method, such as discretisation and consistency errors are discussed in Section 3.2.

For fluid flows, two main SPH frameworks exist. The first is the weakly compressible SPH (WCSPH), which is based on the compressible form of the Navier-Stokes equations where the density and pressure are related by an equation of state. This

is the approach used originally for fluid simulations in [99]. In this framework, the Mach number is chosen in such a way that limits compressibility to a low value, say under 1%, to simulate incompressible flows without enforcing an incompressibility constraint. The second framework is a truly incompressible formulation [31], referred to as ISPH, which explicitly tracks the free surface and solves a pressure Poisson equation over the fluid boundaries to account for incompressibility (i.e. $\nabla \cdot \mathbf{u} = 0$).

The WCSPH does not require any special treatment of the free surface or the solution of a Poisson equation, which is the main drawback of ISPH. Further, WCSPH has been successfully and efficiently adapted to massively parallel codes [19, 28, 62, 126], while this work is still ongoing for ISPH [57]. Due to these considerations, it was decided to focus only on the WCSPH formulation.

Early uses of WCSPH for fluid flows can be traced back to the study free-surface flows such as breaking waves [103, 125]. Currently WCSPH is being used for many applications where grid-based methods are limited such as complicated fluid-structure interactions with floating objects [17], coastal sediment transport [88], and Pelton turbines simulations [89] to name but a few.

2.3.1 SPH for sloshing problems

The WCSPH method should in principle be perfectly suited for the sloshing problem in partially filled tanks. The highly non-linear behaviour and discontinuities of the free surface can naturally be captured by the method without needing any special treatment [22], greatly reducing the complexity of the simulation compared to alternative CFD methods.

SPH has been extensively used for the simulation of sloshing, commonly in relation to marine engineering and liquid natural gas (LNG) tankers [35, 132]. The majority of SPH simulations for sloshing targets short duration or steady-state cases in two-dimensional rectangular tanks. This geometry is more convenient for both the experimental and numerical study of slamming, roof impacts and air trapping, which have all been examined. Examples of such research are found in references [24, 34, 73, 133, 135]. Similar examples in three dimensions can be found in [117].

Advancements in computer hardware and the development of massively parallel SPH simulation tools have permitted to attempt simulations of sloshing in three-dimensions involving complex industrial geometries. Examples however are few and far between. Longshow and Rogers [85] simulated an automotive fuel cell under

high accelerations in three dimensions. The authors reported good agreement in terms of bulk fluid kinematics and reasonable agreement in terms of fluid levels. Cercós-Pita [20] simulated sloshing in a nuclear reactor fuel under earthquake induced motions with three million particles over twenty seconds. This simulation however was done as a proof of concept and not validated against experimental data. The use of SPH to simulate sloshing of real engineering geometries in three dimensions is only emerging and has yet to show its full potential. One of the aims of this thesis is to investigate its capabilities for such complex cases.

2.3.2 Massively parallel SPH

A major drawback of the SPH method is its high computational cost [27]. Unlike grid-based methods where there is a limited number of connected nodes on which to perform calculation, the weighted average interpolations in SPH require each particle to interact with all particles in a compact support domain. In two dimensions the number of interactions between particles is around 20 to 30 depending on the weighted average function chosen and its properties, and around 70 to 120 in three dimensions (see Section 3.1.4). Due to the number of particles required for the simulations of many real physics and engineering problems, SPH simulations can become extremely expensive. Luckily, the Lagrangian nature of SPH (i.e. the lack of an advection term), as well as the explicit time-stepping schemes commonly employed, make SPH a strong candidate for efficient parallelisation. The main difficulty in parallelising SPH is that very fact that particles follow the flow. Combined with the lack on connectivity between the particles mentioned above, neighbor searching is required after each update of the particle positions in order to find the surrounding particles within the weighted average domain. However, advance neighbor searching, for instance as described in reference [39], requires only a small computational cost compared to the total run time of the simulation, accounting for only around 1–5%.

Parallel SPH codes were used as early as 1997 for astronomical problems [32], and later applied also to fluid flows with codes such as Parallel SPHysics [126]. Early implementations of particle methods on graphical processing units (GPUs) can be tracked back to reference [4] where the main code was run on the central processing unit (CPU) and the more arithmetically expensive particle interactions run on the GPU. The SPH method was also later implemented on the GPU for computer graphics [58] where other parts of the code (and not just the particle interactions) were also run exclusively on the GPU. These codes were developed prior

to the release of either CUDA in 2007 [109] or OpenCL in 2009 [56] and instead used OpenGL [74], which is a cross-platform application programming interface (API) for rendering vector graphics and therefore practically inefficient for general purpose computations. H  rault et al. [60] ported SPHysics [51] to GPUs using CUDA, which allowed for the entire simulation to run on the GPU and showed significant speed-ups. Oger et al. [112] used a hybrid CPU-GPU approach using hybrid multi-core parallel programming [36] for the GPU acceleration and message passing interface (MPI) [48] to communicate between computing nodes. A similar approach was taken by Cerc  s-Pita et al. using OpenCL [19]. Both of these codes can run on a wider variety of hardware. Lastly, DualSPHysics [28] was developed as both a CPU only or GPU only code with the CPU version using OpenMP [10], which is a commonly used compiler directive API used for multi-core CPUs, and the GPU version written in CUDA. DualSPHysics is based on SPHysics [51] with a defined goal of achieving the maximum efficiency of the hardware (CPU or GPU) and is one of the most common open-source SPH codes still under active development at the time of writing. A discussion on the GPU hardware architecture and programming on Nvidia GPUs, which has briefly been mentioned here, is given in Section 4.1.

Chapter 3

Smoothed particle hydrodynamics (SPH)

This chapter describes the basic formulation of SPH for weakly compressible flows (WCSPH), with an emphasis on boundary conditions and dissipation methods, the addition of external forces in a moving frame of reference and the time integration scheme. Post-processing techniques for analysing SPH simulations are also described.

3.1 Basic SPH formulation

This section describes the basic WCSPH formulation in the absence of any boundaries and dissipation. It introduces the governing equations for weakly compressible flow in a Lagrangian formulation and describes their discretisation using the SPH method where the SPH operators are derived in a general form which ensures their consistency.

3.1.1 Governing equations of weakly compressible flow

The Lagrangian form of the compressible isentropic Navier-Stokes equations is

$$\begin{aligned}\frac{D\rho}{Dt} &= -\rho\nabla\cdot\mathbf{v} = M \\ \frac{D\mathbf{v}}{Dt} &= \frac{1}{\rho}\nabla\boldsymbol{\sigma} + \mathbf{f}_e = \mathbf{F} \\ \frac{D\mathbf{x}}{Dt} &= \mathbf{v}\end{aligned}\tag{3.1}$$

where D/Dt denotes the material derivative, ρ is the density, $\mathbf{x}$ is the position vector of a fluid particle, $\mathbf{v}$ is the velocity vector and $\mathbf{f}_e$ represents the body force per unit mass. The stress tensor $\boldsymbol{\sigma}$ is given by

$$\boldsymbol{\sigma} = -P\mathbf{I} + \eta \left\{ -\frac{2}{3}(\nabla \cdot \mathbf{v})\mathbf{I} + (\nabla \mathbf{v} + \nabla \mathbf{v}^T) \right\}$$

where $\mathbf{I}$ is the identity tensor, η denotes dynamic viscosity, and the pressure P is calculated using a suitable equation of state. In WSPH a commonly used equation of state is Tait's constitutive equation

$$P = \frac{\rho_0 c_0^2}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right] \quad (3.2)$$

where ρ_0 is a reference density, c_0 is the reference speed of sound and γ is the polytropic constant. For water it is common in SPH to use $\rho_0 = 1000\text{kg/m}^3$, $\gamma = 7$ and c_0 is chosen to have a nominal low Mach number, for instance $Ma < 0.1$ where $Ma = v/c_0$ and v is the local velocity magnitude. One way of choosing c_0 is by setting $c_0 = Cv_{max}$ with v_{max} an estimated maximum velocity of the system and C a coefficient normally taken between 10 – 30. In this way, the higher the value of the coefficient, the smaller the Mach number, and hence the less compressible the fluid becomes. Using this approach, c_0 is significantly lower than the physical speed of sound. Unfortunately this compromise is required in order to avoid an unfeasible number of time-steps due to the stability time-step restriction (see Section 3.7) and in turn limits the value of the Mach number.

3.1.2 SPH discretisation

To numerically approximate the field equations (3.1), the domain is discretised into Lagrangian integration points, commonly referred to as particles, and their properties are evaluated and advanced in time using the SPH methodology.

The SPH method is based upon the integral identity for the value of a function at a particle located at $\mathbf{x}$

$$f(\mathbf{x}) = \int_{\mathbb{R}^3} f(\mathbf{x}') \delta(\mathbf{x} - \mathbf{x}') d\mathbf{x}' \quad (3.3)$$

where f is a sufficiently smooth function in $\mathbb{R}^3$ and δ is the Dirac delta function

$$\int_{\mathbb{R}^3} \delta(\mathbf{x}) = 1; \quad \delta(\mathbf{x}) = \begin{cases} \infty, & \mathbf{x} = 0 \\ 0, & \mathbf{x} \neq 0. \end{cases} \quad (3.4)$$

A numerical approximation of this expression can be obtained by substituting the Dirac delta function by an appropriate smoothing *kernel function*, W , as

$$\langle f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}'. \quad (3.5)$$

Here Ω is a finite region, usually a sphere (or a circle in two dimensions), surrounding the particle, and h is a characteristic length, referred to as the smoothing length. The properties of the smoothing kernel function are described in section 3.1.4.

Replacing $\mathbf{x}$ and $\mathbf{x}'$ with discrete points $\mathbf{x}_a$ and $\mathbf{x}_b$ and using the notation $W_{ab} = W(\mathbf{x}_a - \mathbf{x}_b, h)$, a particle-based approximation of equation (3.5) is given by the numerical quadrature as

$$\langle f(\mathbf{x}_a) \rangle \approx \sum_{b=1}^N V_b f(\mathbf{x}_b) W_{ab} \quad (3.6)$$

where a denotes the selected particle, b denotes the neighbouring particles in the support domain Ω and N is the number of particles within the support domain. The volume of the particle, V_b , is defined as

$$V_b = \frac{m_b}{\rho(\mathbf{x}_b)} \quad (3.7)$$

where m_b is the mass of particle b and $\rho(\mathbf{x}_b)$ is the density of the fluid at $\mathbf{x}_b$. This is illustrated in Figure 3.1.

For instance, the density ρ_a associated with a particle can be obtained using the discrete SPH approximation as

$$\rho_a = \sum_{b=1}^N m_b W_{ab}. \quad (3.8)$$

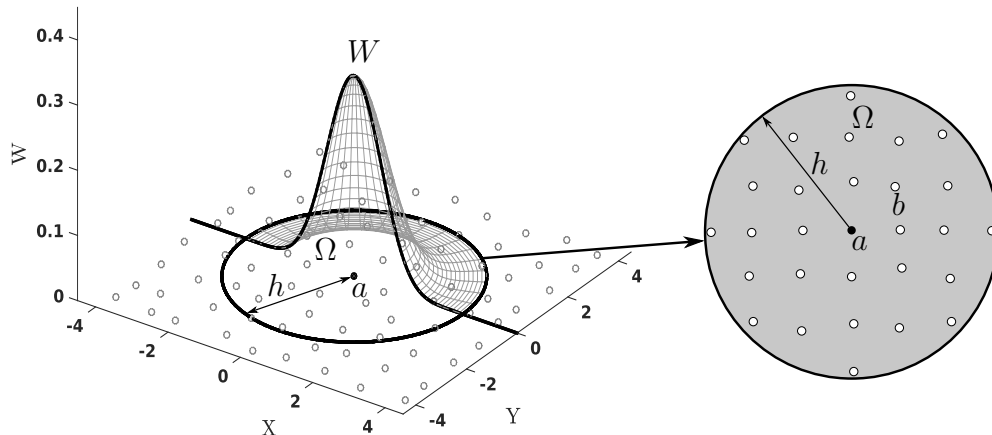


Figure 3.1: The particle-based SPH approximation and notation.

The approximation of the gradient and divergence SPH operators can be obtained in terms of those of the kernel as follows.

Using the Gauss theorem, dropping the integral over the smoothing kernel function's support domain, $\partial\Omega$, and choosing an even smoothing kernel function, i.e.

$\nabla_{\mathbf{x}'}W(\mathbf{x} - \mathbf{x}', h) = -\nabla_{\mathbf{x}}W(\mathbf{x} - \mathbf{x}', h)$, the gradient of a scalar function can be written as

$$\langle \nabla f(\mathbf{x}) \rangle = \int_{\Omega} \nabla_{\mathbf{x}'} f(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3.9)$$

$$= \int_{\partial\Omega} f(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) \mathbf{n} dS - \int_{\Omega} f(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3.10)$$

$$= \int_{\Omega} f(\mathbf{x}') \nabla_{\mathbf{x}'} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}'. \quad (3.11)$$

Similarly, the divergence of a vector function is given by

$$\langle \nabla \cdot \mathbf{f}(\mathbf{x}) \rangle = \int_{\Omega} [\nabla_{\mathbf{x}'} \cdot \mathbf{f}(\mathbf{x}')] W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3.12)$$

$$= \int_{\partial\Omega} \mathbf{f}(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) \cdot \mathbf{n} dS - \int_{\Omega} \mathbf{f}(\mathbf{x}') \cdot \nabla_{\mathbf{x}'} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3.13)$$

$$= \int_{\Omega} \mathbf{f}(\mathbf{x}') \cdot \nabla_{\mathbf{x}'} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}'. \quad (3.14)$$

Using the notation $\nabla_a W_{ab} = \nabla W(\mathbf{x}_a - \mathbf{x}_b, h)$ where the differentiation is carried out with respect to $\mathbf{x}_a$, a particle-based approximation of equations (3.11) and (3.14)

is given by

$$\langle \nabla f(\mathbf{x}) \rangle \approx \sum_b V_b f(\mathbf{x}_b) \nabla_a W_{ab} \quad (3.15)$$

and

$$\langle \nabla \cdot \mathbf{f}(\mathbf{x}) \rangle \approx \sum_b V_b \mathbf{f}(\mathbf{x}_b) \cdot \nabla_a W_{ab} \quad (3.16)$$

respectively.

From here on, the approximation notation “ $\langle f(\mathbf{x}) \rangle \approx$ ” is replaced by “ $f(\mathbf{x}) =$ ” in the understanding that in the SPH context these are approximations.

3.1.3 Governing equations in semi-discrete form

The governing equations (3.1) can now be described in a semi-discrete form by replacing the continuous variables with discrete ones and applying the SPH approximations of the divergence and gradient operators. Dropping the viscosity term for simplicity, they are given as

$$\frac{D\rho_a}{Dt} = -\rho_a \nabla \cdot \mathbf{v}_a = \sum_b M_{ab} = M_a \quad (3.17)$$

$$\frac{D\mathbf{v}_a}{Dt} = \frac{1}{\rho_a} \nabla(-P_a) + \mathbf{f}_e = \sum_b \mathbf{F}_{ab} + \mathbf{f}_e = \mathbf{F}_a \quad (3.18)$$

$$\frac{D\mathbf{x}_a}{Dt} = \mathbf{v}_a \quad (3.19)$$

and

$$P_a = \frac{\rho_0 c_0^2}{\gamma} \left[\left(\frac{\rho_a}{\rho_0} \right)^\gamma - 1 \right]. \quad (3.20)$$

Here, denotes M_{ab} the inter-particle density derivative and $\mathbf{F}_{ab}$ the inter-particle force per unit mass. By applying SPH operators to the velocity and pressure terms in the semi-discrete equations (3.17–3.18), the basic WCSPH formulation can then be derived.

3.1.4 Smoothing kernel function and its derivative

A suitable choice of smoothing kernel function, or kernel for short, should have the following properties. For equation (3.5) to be accurate to the first order, the kernel should be equivalent to the Dirac δ function (3.4) in the limit and satisfy the zeroth

and first-order consistency conditions. These are expressed mathematically as

$$\lim_{h \rightarrow 0} W(\mathbf{x} - \mathbf{x}', h) = \delta(\mathbf{x} - \mathbf{x}'), \quad (3.21)$$

$$\int_{\Omega} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' = 1, \quad (3.22)$$

and

$$\int_{\Omega} (\mathbf{x} - \mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' = 0, \quad (3.23)$$

respectively.

The derivation of efficient and consistent SPH operators in equations (3.11) and (3.14) requires the kernel to have compact support, i.e.

$$W(\mathbf{x} - \mathbf{x}', h) = 0; \quad \|\mathbf{x} - \mathbf{x}'\| > \kappa h \quad (3.24)$$

where κ is a constant, and to be an even symmetric function, namely

$$W(\mathbf{x} - \mathbf{x}', h) = W(\mathbf{x}' - \mathbf{x}, h) \quad (3.25)$$

$$\nabla_{\mathbf{x}'} W(\mathbf{x} - \mathbf{x}', h) = -\nabla_{\mathbf{x}} W(\mathbf{x} - \mathbf{x}', h). \quad (3.26)$$

The kernel should also be positive, monotonically decreasing and sufficiently smooth. The need for these properties is discussed at length in reference [83].

Kernel functions are often written in the form

$$W(\mathbf{x} - \mathbf{x}', h) = \frac{\alpha_d}{h^d} f(q) \quad (3.27)$$

where α_d is a scaling factor to ensure the consistency condition (3.22), d is the spatial dimension and q is a dimensionless number, representing a unit length, given by

$$q = \frac{\|\mathbf{x} - \mathbf{x}'\|}{h}.$$

Using this notation, a derivative of the kernel function with respect to $\mathbf{x}$ can be obtained as

$$\nabla W(\mathbf{x} - \mathbf{x}', h) = \frac{\alpha_d}{h^d} \nabla f(q) = \frac{\alpha_d}{h^d} \frac{df(q)}{dq} \nabla q = \frac{\alpha_d}{h^{d+1}} \frac{df(q)}{dq} \frac{\mathbf{x} - \mathbf{x}'}{\|\mathbf{x} - \mathbf{x}'\|}. \quad (3.28)$$

Common choices for kernel functions are the Gaussian and B-spline functions. More recently the Wendland kernels [148] have become popular. These kernels have the advantage of being defined by a single algebraic equation (unlike the B-spline function) while also having compact support [146]. The choice of the kernel function has implications on both the accuracy of the reproduced function and on the computational costs, with a good kernel striking a balance between both.

The quintic Wendland kernel satisfies all the kernel requirements and has been identified as an ideal candidate for SPH [33]. Therefore it has been exclusively used in all the work presented in this thesis. It is given by the expression

$$W(\mathbf{x} - \mathbf{x}', h) = \frac{\alpha_d}{h^d} (2q + 1) \left(1 - \frac{q}{2}\right)^4; \quad 0 \leq q \leq 2 \quad (3.29)$$

and its derivative by

$$\nabla W(\mathbf{x} - \mathbf{x}', h) = \frac{\alpha_d}{h^{d+1}} \left[-5q \left(1 - \frac{q}{2}\right)^3 \right] \frac{\mathbf{x} - \mathbf{x}'}{\|\mathbf{x} - \mathbf{x}'\|}; \quad 0 \leq q \leq 2 \quad (3.30)$$

where $\alpha_d = \frac{7}{4\pi}$ in two-dimensional (2D) space and $\alpha_d = \frac{21}{16\pi}$ in three-dimensional space (3D), noting that both functions are 0 for $q > 2$. To better visualise this kernel, a plot of the two-dimensional function and its derivative are given in Figure 3.2.

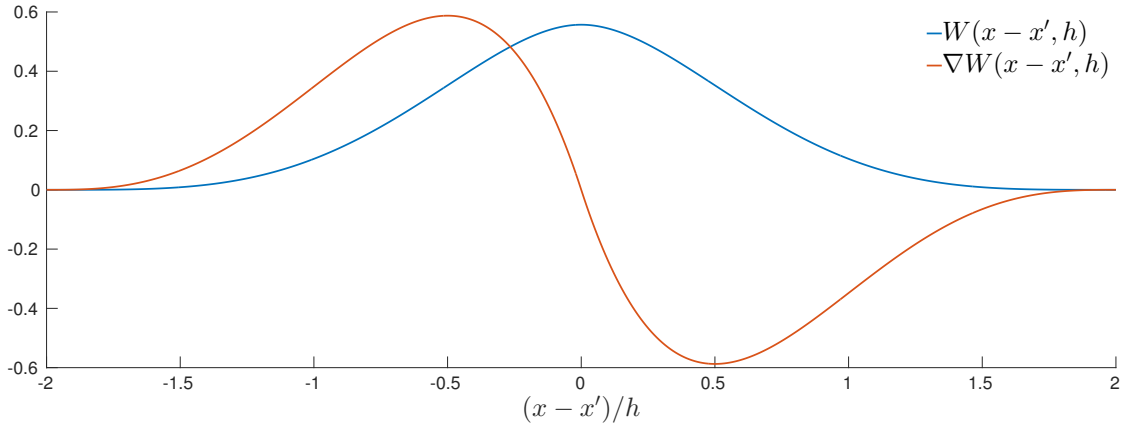


Figure 3.2: The two-dimensional Wendland kernel and its derivative over the compact support domain $|q| < 2$.

The smoothing length, h , is commonly chosen as a factor of the particle spacing Δx , i.e.

$$h = w_h \Delta x \quad (3.31)$$

where w_h is a coefficient with values usually taken in the range $1 \leq w_h \leq 2$. For the Wendland kernel and kernels of a $2h$ support, common choices are $w_h = 1.3$ or $w_h = 1.5$ which respectively give 21 or 29 particles in the 2D kernel support domain and 74 or 114 in 3D. Kernels with a support of $3h$, such as the quintic spline, usually use $w_h = 1$ [3] to keep the size of the support domain from being too large.

A discussion of kernel functions used for SPH can be found in reference [146] and a more comprehensive study in references [33,64,81]. It is important to note that the first-order consistency condition requires the particles to be evenly distributed and for the kernel support to be filled evenly. However, in real SPH applications, such perfect conditions rarely exists, leading to the introduction of inconsistency related errors. This is explained in more detail in section 3.2.

3.1.5 General formulation of the gradient and divergence operators

The first-order derivative operators (3.15) and (3.16), although the simplest, are not consistent. Notably, the gradient operator (3.15) in the momentum conservation equation (3.18) is not anti-symmetric which leads to $\mathbf{F}_{ab} \neq -\mathbf{F}_{ba}$ in contradiction with Newton's third law. Further, these operators do not have the skew-adjoint properties of the continuous operators which is vital for the method to be variationally consistent [13,146].

Following [146], a general form most commonly used in the weakly compressible SPH method is introduced, which covers all the *standard* SPH formulations.

Starting with

$$\nabla f(\mathbf{x}) = \nabla \left(\rho^k \frac{f(\mathbf{x})}{\rho^k} \right) = \rho^k \nabla \frac{f(\mathbf{x})}{\rho^k} + \frac{f(\mathbf{x})}{\rho^k} \nabla \rho^k, \quad (3.32)$$

$$\nabla \cdot \mathbf{f}(\mathbf{x}) = \nabla \cdot \left(\frac{1}{\rho^k} \rho^k \mathbf{f}(\mathbf{x}) \right) = \frac{1}{\rho^k} (\nabla \cdot \rho^k \mathbf{f}(\mathbf{x})) - \frac{\mathbf{f}(\mathbf{x})}{\rho^k} \cdot \nabla \rho^k \quad (3.33)$$

where k is an arbitrary coefficient, and applying the standard SPH gradient (3.15)

and divergence (3.16) operators, the general operators are obtained as

$$\begin{aligned}
\nabla f(\mathbf{x}) &= \rho_a^k \sum_b V_b \frac{f(\mathbf{x}_b)}{\rho_b^k} \nabla_a W_{ab} + \frac{f(\mathbf{x}_a)}{\rho_a^k} \sum_b V_b \rho_b^k \nabla_a W_{ab} \\
&= \sum_b V_b \rho_a^k \frac{f(\mathbf{x}_b)}{\rho_b^k} + \frac{f(\mathbf{x}_a)}{\rho_a^k} \rho_b^k \nabla_a W_{ab} \\
&= \sum_b V_b \frac{\rho_b^{2k} f(\mathbf{x}_a) + \rho_a^{2k} f(\mathbf{x}_b)}{(\rho_a \rho_b)^k} \nabla_a W_{ab},
\end{aligned} \tag{3.34}$$

$$\begin{aligned}
\nabla \cdot \mathbf{f}(\mathbf{x}) &= \frac{1}{\rho_a^k} \sum_b V_b \rho_b^k \mathbf{f}(\mathbf{x}_b) \cdot \nabla_a W_{ab} - \frac{\mathbf{f}(\mathbf{x}_a)}{\rho_a^k} \cdot \sum_b V_b \rho_b^k \nabla_a W_{ab} \\
&= \frac{1}{\rho_a^k} \sum_b V_b \rho_b^k (\mathbf{f}(\mathbf{x}_b) - \mathbf{f}(\mathbf{x}_a)) \cdot \nabla_a W_{ab} \\
&= -\frac{1}{\rho_a^{2k}} \sum_b V_b (\rho_a \rho_b)^k (\mathbf{f}(\mathbf{x}_b) - \mathbf{f}(\mathbf{x}_a)) \cdot \nabla_a W_{ab}.
\end{aligned} \tag{3.35}$$

Using this formulation, the gradient operator is symmetric, while the divergence operator is anti-symmetric. This is consistent with Newton's third law as the force per unit mass, given in terms of the acceleration, uses only the gradient operator. This form also guarantees the discrete operators are skew-adjoint and variationally consistent, as will be explained in detail in Section 3.1.7.

It should be noted that the divergence operator reproduces a constant field (i.e. $\nabla \cdot \mathbf{1} = 0$). Unfortunately, the gradient operator does not and enforcing this property leads to the loss of variational consistency. However, this formulation is the most favourable one according to reference [146].

3.1.6 Governing equations in WCSPH formalism

Replacing the SPH operators in the semi-discrete governing equations (3.17 – 3.19) with (3.34) and (3.35), and rearranging the terms, a general formulation of the continuity and momentum conservation equations can be obtained as

$$\frac{D\rho_a}{Dt} = \frac{1}{\rho_a^{2k-2}} \sum_b m_b (\rho_a \rho_b)^{k-1} (\mathbf{v}_a - \mathbf{v}_b) \cdot \nabla_a W_{ab}, \tag{3.36}$$

$$\frac{D\mathbf{v}_a}{Dt} = - \sum_b m_b \left(\frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} \right) \nabla_a W_{ab} + \mathbf{f}_e \tag{3.37}$$

where k usually taken to be 0 or 1. These options are referred to as the “continuity” ($k = 0$) and “direct density” ($k = 1$) formulations [45]. These can be written as

$$\frac{D\rho_a}{Dt} = \rho_a \sum_b V_b (\mathbf{v}_a - \mathbf{v}_b) \cdot \nabla_a W_{ab} \quad (3.38)$$

$$\frac{D\mathbf{v}_a}{Dt} = -\frac{1}{\rho_a} \sum_b V_b (P_a + P_b) \nabla_a W_{ab} + \mathbf{f}_e \quad (3.39)$$

for the continuity formulation and

$$\frac{D\rho_a}{Dt} = \sum_b m_b (\mathbf{v}_a - \mathbf{v}_b) \cdot \nabla_a W_{ab} \quad (3.40)$$

$$\frac{D\mathbf{v}_a}{Dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla_a W_{ab} + \mathbf{f}_e \quad (3.41)$$

for the direct density formulation. The main difference between these formulations is that the continuity formulation uses $\rho_a V_b$ multiplied by the velocity difference while the other uses m_b in the continuity equation (3.38). This can have a significant effect when large density fluctuations occur, so that having a mass term of the form $\rho_a V_b$ is preferable [67]. The naming convention of these formulations is used due to an alternative approach in deriving the discrete SPH operators which is not followed here, but will be briefly described. The continuity formulation can be obtained by applying the SPH divergence operator in the continuity equation and then modifying it to obtain the correct divergence of a constant field. The direct density formulation uses the SPH discretisation of the density as in equation (3.8) and performs the divergence of the discrete equation [45]. Here, this naming convention is used only to offer a more intuitive way of distinguishing the formulation than using $k = 0$ and $k = 1$.

Following reference [2], an alternative formulation of the momentum equation can be derived where the pressure is calculated by an inter-particle average, given as

$$\frac{D\mathbf{v}_a}{Dt} = -\frac{1}{m_a} \sum_b ((V_a^2 + V_b^2) \tilde{p}_{ab}) \nabla_a W_{ab} + \mathbf{f}_e, \quad (3.42)$$

where $\tilde{p}_{ab}$ is an inter-particle averaged pressure calculated as

$$\tilde{p}_{ab} = \frac{\rho_b p_a + \rho_a p_b}{\rho_a + \rho_b}. \quad (3.43)$$

This formulation is reported to be more robust in combination with the continuity

equation (3.38) for multi-fluid simulations due to the average pressure term being more uniform across the phases interface [67] .

To conclude, the basic WCSPH formulation consists of equations (3.17–3.19) with either the direct density formulation (3.40–3.41), the continuity formulation (3.38–3.39), the inter-particle average formulation (3.38) and (3.42), or any other formulation for the SPH operators.

3.1.7 Lagrangian derivation of the SPH operators

The Lagrangian nature of SPH and the discretisation into particles represent a physical system as a system of particles. Using this representation, it is possible to exploit the Hamiltonian principle of classical mechanics [72] to obtain SPH operators which are variationally consistent, so that they conserve all the properties of the physical system. In this derivation the total energy defined by the Lagrangian and using the continuity equation (3.36) with the divergence operator in equation (3.35), is used to obtain the momentum equation (3.37) with the gradient operator in equation (3.34).

The total energy in a system representing fluid particles can be given by the Lagrangian as

$$L = \int \left(\frac{1}{2} \rho v^2 - \rho u \right) dV \quad (3.44)$$

where v is the velocity and u is the internal energy per unit mass. Writing the equation in terms of particles with a fixed mass m_b we get

$$L = \sum_b m_b \left(\frac{1}{2} v_b^2 - u_b(\rho_b, s_b) \right) \quad (3.45)$$

where the volume element ρdV is replaced with the mass of a particle m_b , and the internal energy of particle as u_b is given in terms of the density of the particle ρ_b and any other constitutive parameters s_b .

Following a general derivation similar to that of Bonet and Lok [13], the stationary action representing a minimum of L is calculated via a small variation with respect to the particle position $\delta \mathbf{x}$ leading to the Euler-Lagrange equation

$$\delta \int L dt = \int \delta L dt = \int \left(\frac{\partial L}{\partial \mathbf{v}} \delta \mathbf{v} + \frac{\partial L}{\partial \mathbf{x}} \delta \mathbf{x} \right) dt = 0. \quad (3.46)$$

In the reversible adiabatic case, the internal energy is a function of the fluid's density

only. Calculating the variation δL in equation (3.45) we get

$$\delta L = m_a \mathbf{v}_a \cdot \delta \mathbf{v}_a - \sum_b m_b \frac{\partial u_b}{\partial \rho_b} \frac{\delta \rho_b}{\delta \mathbf{x}_a} \delta \mathbf{x}_a \quad (3.47)$$

where $\delta \rho_b$ can be found using the continuity equation (3.36) as

$$\delta \rho_b = \rho_b \nabla \cdot \frac{\delta \mathbf{x}_b}{\delta t} \delta t = \rho_b \nabla \cdot \delta \mathbf{x}_b = \frac{1}{\rho_b^{2k-2}} \sum_c m_c (\rho_b \rho_c)^{k-1} (\delta \mathbf{x}_b - \delta \mathbf{x}_c) \cdot \nabla_b W_{bc}. \quad (3.48)$$

Under these assumptions, the partial derivative of the internal energy can be derived as [14]

$$\frac{\partial u_b}{\partial \rho_b} = \frac{P_b}{\rho_b^2}. \quad (3.49)$$

Substituting equations (3.48) and (3.49) into equation (3.47) we have

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} &= - \sum_b m_b \frac{P_b}{\rho_b^2} \frac{1}{\rho_b^{2k-2}} \sum_c m_c (\rho_b \rho_c)^{k-1} (\delta_{ba} - \delta_{ca}) \nabla_b w_{bc} \\ &= - \sum_b \sum_c m_b m_c \frac{P_b}{\rho_b^{2k}} (\rho_b \rho_c)^{k-1} (\delta_{ba} - \delta_{ca}) \nabla_b w_{bc} \\ &= - \sum_c m_a m_c \frac{P_a}{\rho_a^{2k}} (\rho_a \rho_c)^{k-1} \nabla_a w_{ac} + \sum_b m_b m_a \frac{P_b}{\rho_b^{2k}} (\rho_b \rho_a)^{k-1} \nabla_b w_{ba} \end{aligned} \quad (3.50)$$

where δ_{ij} is the Kronecker delta function. As the subscripts are arbitrary, it is possible to write (3.50) as

$$\frac{\partial L}{\partial \mathbf{x}} = - \sum_b m_a m_b (\rho_a \rho_b)^{k-1} \left(\frac{P_a}{\rho_a^{2k}} \nabla_a W_{ab} - \frac{P_b}{\rho_b^{2k}} \nabla_b W_{ba} \right). \quad (3.51)$$

By invoking the symmetry property of the kernel function, i.e. $\nabla_a W_{ab} = -\nabla_b W_{ba}$ (see section 3.1.4) this equation can be written as

$$\frac{\partial L}{\partial \mathbf{x}} = - \sum_b m_a m_b \frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} \nabla_a W_{ab}. \quad (3.52)$$

Substituting (3.52) into (3.46) and integrating the velocity term by parts we get

$$\int \left(-m_a \frac{d\mathbf{v}_a}{dt} - m_a \sum_b m_b \frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} \nabla_a W_{ab} \right) \delta \mathbf{x}_a dt = 0 \quad (3.53)$$

given that $\delta \mathbf{x}_a$ is arbitrary, we can finally rearrange as

$$\frac{d\mathbf{v}_a}{dt} = - \sum_b m_b \left(\frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} \right) \nabla_a W_{ab} \quad (3.54)$$

which is identical to (3.37). Hence, using such operators which conserve the Hamiltonian ensure the scheme is variationally consistent.

3.2 Spatial numerical errors in WCSPH

SPH, like any computational method, suffers from various approximation and numerical errors. These include errors relating to the continuous and discrete numerical approximations, the loss of the kernel consistency properties and the loss of conservation properties, which are closely linked together. However, being a relatively young method, SPH still lacks a consistent mathematical theory for stability and convergence [147], making a rigorous evaluation of the numerical errors a very difficult task. A brief discussion providing a general characterisation of some of the many numerical issues, and where relevant suggestions on how to address them, is given in the following.

Numerical errors relating to the free-surface and solid wall boundary conditions are described in Sections 3.3 and 3.4 respectively, and other sources of numerical errors, as well as the means of numerically dissipating them, in Section 3.5.

3.2.1 Discretisation errors

Space-related discretisation errors in SPH appear from two sources, the continuous approximation (3.5) and the discrete approximation (3.6), each characterised by a different (and contradictory) length scale [146]. The continuous approximation is characterised by the smoothing length h , which is a measure of the radius of the compact support of the kernel, and is more accurate as $h \rightarrow 0$. The discrete approximation is characterised by the ratio $\frac{\Delta x}{h}$, which is a measure of the number of particles inside the kernel support, and is more accurate as $\frac{\Delta x}{h} \rightarrow \infty$. According to reference [116], second-order convergence can be obtained when the continuous approximation error is dominant, such as the case when h is large and $\frac{\Delta x}{h}$ is small. However, when keeping $\frac{\Delta x}{h}$ constant (as this is normally the case), decreasing h results in the SPH operators having, at best, an error limited by the discretisation

approximation. This duality of spatial parameters makes any error estimation in SPH hard to obtain, even without taking into account boundary conditions, the truncated kernel at the free surface, or time-integration related errors. The process of estimating the overall error of a SPH simulation for real engineering problems is, at the time of writing, not fully understood.

3.2.2 Consistency errors

Other than the unavoidable discretisation errors discussed above, probably the most prevalent issue in the SPH method is the kernel consistency error. This problem occurs wherever the consistency conditions of the kernel function (3.22 – 3.23) are not met, such as in the vicinity of the free surface or wall boundaries [22,75] or when the particle distribution is not uniform [144].

Non-uniformity of particles can occur for various reasons. One of them is the Lagrangian nature of SPH, which means that particles follow the streamlines rather than keeping a uniform spatial distribution. Another is the infamous “tensile instability” [100]. This is a phenomenon where the pressure of a particle is no longer positive due to the density being lower than its initial value when the pressure is calculated by the equation of state (3.20). This results in particle attraction rather than mutual repulsion.

Different variants of WCSPH have been proposed to deal with the problem of kernel consistency, or its underlying causes. The corrected SPH scheme (CSPH) [11] applies a linear correction term calculated by the moving least-squares method to the kernel and kernel gradient. This approach greatly improves the kernel approximation for non-uniform particle distributions and regions where the kernel’s compact support is not full. Unfortunately, this scheme is complicated to implement and is computationally expensive. A more recent scheme is CRKSPH [118] which is claimed by the authors to be both consistent and conservative. This scheme generates a modified kernel function for each particle depending on the distribution of its surrounding particles. However, it has yet to be validated for cases with solid boundaries and is not, at the time of writing, suitable for real engineering problems.

A different approach to restore consistency is to artificially shift the particles to obtain a more uniform distribution. This approach can be traced to XSPH [99] where particles are shifted in order to smooth the velocity field. A more recent shifting method is described in reference [150], where particles are slightly shifted along the streamlines. This method was further improved to include Fick’s law of

diffusion and a free-surface correction in reference [80]. However, it is important to note that even the best of these shifting methods cannot exactly conserve momentum due to the artificial shift in positions.

3.3 Free-surface boundary conditions in WCSPH

One of the main advantages of the WCSPH for modelling free-surface problems comes from its Lagrangian nature. Since the particles are advected with the flow, the description of the free surface, no matter how complicated, is the result of their motion without any need to explicitly track the interface. While this is the common practice for WCSPH practitioners, careful consideration of the effects of the free surface have to be taken into account. The kernel support at the free surface is truncated, leading to a loss of the consistency condition of the kernel (3.22) and hence numerical errors. This is illustrated in Figure 3.3. Further, the validity of the free-surface boundary conditions should be verified.

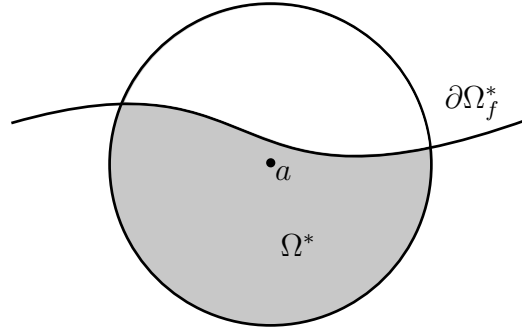


Figure 3.3: Kernel function's compact support in the vicinity of a free surface. Here Ω^* denotes the truncated support domain (grey area) and $\partial\Omega_f^*$ denotes the free surface.

In a landmark paper by Colagrossi et al. [22] a theoretical foundation which, at least on the continuous level, justifies the use of the basic WCSPH formulation for simulating free-surface flows has been laid out. This paper investigates both the kinematic and dynamic free-surface boundary conditions.

The kinematic condition, which states that any fluid particle on the free surface, denoted $\partial\Omega_f^*$, will remain on it, i.e.

$$\mathbf{v} \cdot \mathbf{n}_f = \mathbf{V}_{\partial\Omega_f^*} \cdot \mathbf{n}_f$$

where $\mathbf{n}_f$ is the normal vector, and $\mathbf{V}_{\partial\Omega_f^*}$ is the free-surface velocity. This condition

is implicitly satisfied by the Lagrangian nature of SPH on the continuous level, and hence approximately satisfied in the discretised WCSPH formulation.

The dynamic free-surface boundary condition in the absence of surface tension states that the pressure is continuous across the interface. This means that the pressure on the free surface is equal to the atmospheric pressure. Assuming the atmospheric pressure is constant, and given the equation of state (3.20), this condition is equivalent to requiring a constant density equal to the reference density across the interface. This leads to the continuous form of the continuity equation (3.1) to be reduced to

$$\nabla \cdot \mathbf{v} = 0 \quad \text{on } \partial\Omega_f^*.$$

The authors conclude that the variationally consistent terms, such as those derived in Section 3.1.5, approximately satisfy the dynamic free-surface condition without the need for explicitly implementing free-surface terms. However, as a consequence, a WCSPH formulation with a free surface present can be either consistent or satisfy the Hamiltonian conservation (i.e. variationally consistent), but not simultaneously both. The reader is referred to the full article [22] for full details of this conclusion.

It is important to note however that reference [22] only gives some justification for the common practice of not explicitly treating the free surface in the WCSPH, but does not guarantee its absolute validity. The study was limited to the continuous, and not the discrete operators, and did not attempt to investigate higher order terms such as the Laplacian in the presence of the free surface. This means that it is still up to the WCSPH practitioner to judge the validity of the simulation and to take into account the possibility of additional errors that could be generated by this choice.

3.4 Wall boundary conditions

Wall boundary conditions are at the heart of SPH research and are identified as one of the grand challenges of the SPH method set out by the SPH European Research Interest Community (SPHERIC) which acts as an international organisation representing the community of researchers and industrial users of SPH [136].

Solid wall boundaries should prevent particles penetrating the wall, i.e. $\mathbf{v} \cdot \mathbf{n} = 0$ at the wall, denoted $\partial\Omega_b^*$, where $\mathbf{n}$ is the normal to the solid wall. They should also apply a free-slip or no-slip condition depending on whether the fluid is viscous or inviscid respectively. However, a problem arises from the truncation of the kernel support by the solid boundary, which renders kernel consistency assumption (3.22)

false. The completeness of a kernel can be characterised by the integral over the kernel support, which is denoted γ following [13], and is given by

$$\gamma(\mathbf{x}, h) = \int_{\Omega^*} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \quad (3.55)$$

which is illustrated in Figure 3.4. In the case of a full kernel support $\Omega^* = \Omega$ and $\gamma \approx 1$, as per the consistency condition (3.22), while over the truncated support $\Omega^* \neq \Omega$ so that $\gamma \neq 1$ and the consistency condition no longer holds true.

A large range of methods to deal with this deficiency of the SPH method near solid walls have been proposed over the years in the SPH literature and these are reviewed in a general fashion in the following section.

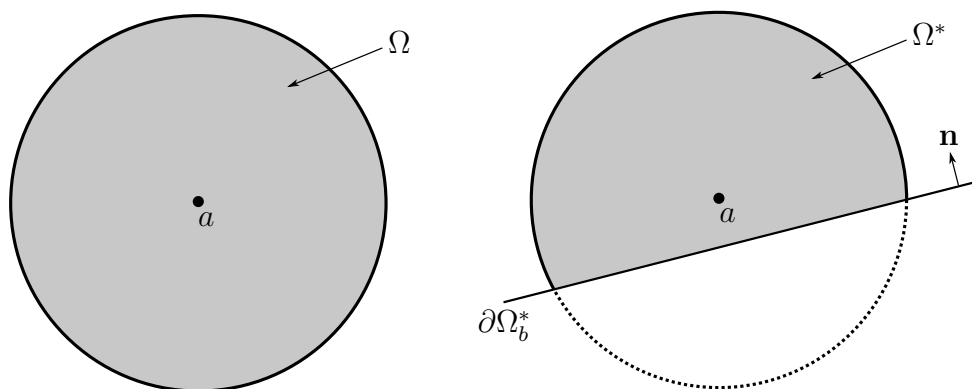


Figure 3.4: Kernel function's compact support for a particle in the bulk of the fluid (left), and in the proximity of a solid boundary (right). Here Ω denotes a full kernel support, Ω^* a truncated kernel support and $\partial\Omega_b^*$ a solid boundary segment.

3.4.1 Wall boundary conditions review

Wall boundary conditions in SPH can generally be categorised into three types [46, 146]: repulsive force boundary conditions (RFBC), semi-analytical boundary conditions (SABC) and ghost particle boundary conditions (GPBC). An illustration of the different types is shown in Figure 3.5.

The RFBC methods use particles on the boundary to apply a force $\mathbf{f}$ normal to the wall on a fluid particle when its kernel region intersects the wall. This force prevents particle penetration, and mimics the fundamental molecular basis of the boundary conditions, i.e. that the atoms of the fluid do not penetrate the solid due to the atomic forces between them. Many formulations have been devised for the normal force such as those proposed in references [99, 102].

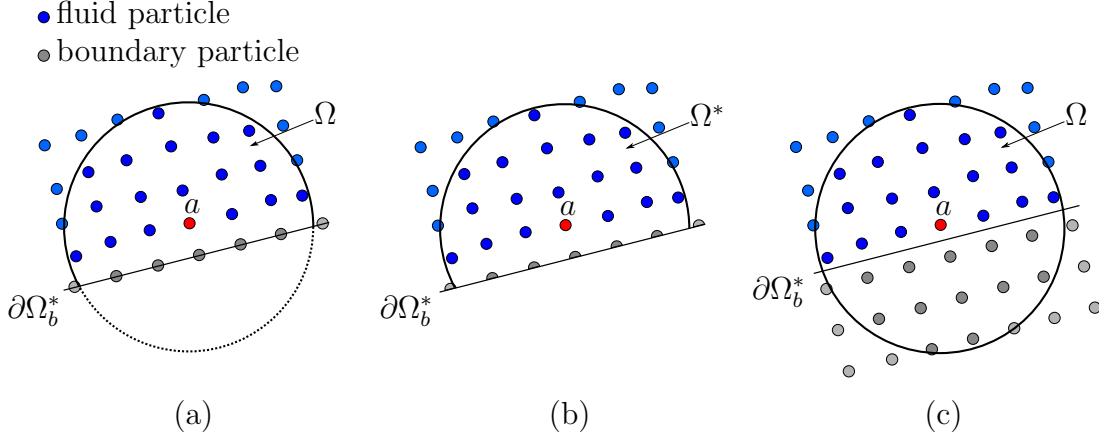


Figure 3.5: Illustration of different wall boundary condition types: (a) Repulsive force boundary conditions (RFBC). Showing the full kernel support Ω and the boundary segment $\partial\Omega_b^*$. The dashes represent the *missing* region of Ω ; (b) Semi-analytical boundary conditions (SABC). Showing the truncated support domain Ω^* . The boundary particles represent the discretisation of the boundary segment $\partial\Omega_b^*$; Fixed ghost particles boundary conditions (GPBC). Showing the full kernel support Ω and the additional boundary particles to ensure Ω is full.

A simple formulation for the RFBC in two-dimensions, taken from reference [127], gives the force as

$$\mathbf{f} = \mathbf{n} R(\delta_n) P(\xi) \varepsilon(z, v_n)$$

where v_n is the velocity of the fluid particle normal to the boundary, ξ is a coordinate along the side of two adjacent boundary particles, δ_n is the distance of the fluid particle from the wall. The repulsion function, R , evaluates a force in terms of the normalised distance from the fluid particle to the wall. This value is then scaled by a function P that incorporates a constant repulsive force for the fluid particles as they travel parallel to the wall and a function ε that adjusts the magnitude of the force according to the elevation above the local still water depth z and the normal velocity v_n . The expressions of these functions can readily be found in [51]. Another method in this group is the dynamic boundary condition [29] which assumes that the wall boundary particles push the fluid particles when approaching. The force that a boundary particle b exerts on a fluid particle a is obtained through a suitable combination of the equations of continuity, momentum and of state to get an expression of the additional acceleration given to the fluid particle as

$$\frac{D\mathbf{v}_a}{Dt} = -2c^2 \frac{W_{ab}}{(W_{ab} + W_0)^2} \nabla_a W_{ab}$$

where $W_0 = W_{aa} = W_{bb}$. This acceleration is then added to the momentum conser-

vation equation (3.18). These methods are generally easy to implement, are computationally cheap, and are readily applicable to complex geometries. However, they do not take into account the error due to the truncated kernel function and thus generate errors in the proximity of the boundaries. These errors can result in non-physical behaviour such as non-uniform gaps between the boundaries and fluid [102] and spurious fluid properties adjacent to the walls [46, 151].

The SABC methods account for the *missing kernel* by integrating over the truncated kernel support domain Ω^* . They were first proposed in reference [75]. For the purpose of this review, only the continuity equation (3.17) is used to explain the basic principle, but a similar derivation can be obtained for the momentum equation, or indeed any other field equation too. The approximation of the density with an incomplete kernel is written as

$$\rho(\mathbf{x}) \int_{\Omega^*} W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' = \int_{\Omega^*} \rho(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}'$$

where the density is evaluated inside the integral. The value of the density will be evaluated only in the fluid domain when the kernel support is truncated. Using equation (3.55), a particle approximation of this expression is

$$\rho(\mathbf{x}_a) \gamma(\mathbf{x}_a, h) = \sum_b m_b W_{ab}.$$

Writing $\gamma_a = \gamma(\mathbf{x}_a, h)$ and taking the material derivative of both sides of this equation we get

$$\frac{D\rho_a}{Dt} \gamma_a + \rho_a \frac{D\gamma_a}{Dt} = \frac{D}{Dt} \left(\sum_b m_b W_{ab} \right) = \sum_b m_b \nabla W_{ab} \cdot (\mathbf{v}_a - \mathbf{v}_b).$$

Suitably rearranging terms, we can reorganise the expression as the continuity equation using the continuity formulation (3.38) with an additional term

$$\frac{D\rho_a}{Dt} = \frac{1}{\gamma_a} \sum_b m_b \nabla W_{ab} \cdot (\mathbf{v}_a - \mathbf{v}_b) - \frac{\rho_a}{\gamma_a} \frac{D\gamma_a}{Dt}$$

The term on the right hand side vanishes, i.e. $\frac{D\gamma_a}{Dt} = 0$, when the kernel region does not intersect the boundary, and acts as a contact force when it does intersect. Analytical and semi-analytical evaluations of γ and $\frac{D\gamma_a}{Dt}$ are described in references [45, 75] and [46, 93] respectively. A more general study of the method which accounts

for the general operators (3.34–3.35) in the governing equations (3.36–3.37) can be found in reference [93]. An initial extension to three dimensions is given in reference [5] where no analytical values are used, or reference [92] which uses pre-defined analytical expressions based on the shape of the boundary. These methods show a lot of promise, but can become very expensive and difficult to implement.

The GPBC methods use additional “fictitious” particles which are placed outside the fluid domain and adjacent to the solid boundary in a manner that ensures the consistency of the kernel. The “fictitious” particles can either be pre-generated and fixed in position, known as *fixed ghost boundary* methods, or generated at every time-step to mirror the fluid particles, known as *mirror particle* methods. These particles are not integrated in time, but instead their values are obtained from interpolation over the adjacent fluid particles or over their mirrored fluid counterparts. To better understand the method, a simplified case for illustration purposes is given. In this example the particles are fixed in position and have a fixed density equal to the reference density ρ_0 . This case represents an overly simplified fixed ghost particle approach, and hence using the standard terminology, the “fictitious” particle will be referred to as a *ghost* particle in the following. As a fluid particle interacts with the ghost particles, the contribution of a ghost particle to the volume of a fluid particle based on the continuity equation (3.17) can be thought of as a volume flux [146], and is given by

$$qV_a = -m_a m_w \frac{\rho_0^{k-1}}{\rho_a^{k+1}} \mathbf{v}_a \nabla_a W_{af} \quad (3.56)$$

where the subscript w corresponds to the ghost particle, and k is an arbitrary constant in accordance with the general formulation of the SPH operators (3.34–3.35). Hence, at least in the weakly-compressible framework, when a fluid particle approaches the solid wall this contribution very slightly decreases the fluid particle’s volume. This in turn increases the density based on the volume to density ratio (3.7) due to the mass being fixed. The pressure then increases according to the equation of state (3.20), which generates a repulsive force normal to the boundary. This over-simplified approach is not likely to provide satisfactory results, however, many more advanced variations of it exist which are amongst the most accurate wall boundary conditions found in the literature to date [3, 90]. These fixed ghost particle methods generally provide a good balance between accuracy, efficiency and ease of implementation are also applicable to complex geometries. In contrast, the mirror particle approach, an example of which can be found in reference [23], can potentially

be more accurate, but it is difficult to mirror particles in complex geometries and they are computationally more expensive. It is however important to note that also the fixed ghost particles methods have some limitations in complex geometries. In particular, when modeling geometries with a large range of scales between parts of the solid boundary, or thin geometries in comparison to the global domain. This is due to the particle spacing being restricted by the number of particle layers required to discretise the thinnest part of the solid boundary and fill the missing kernel support. In the case of the solid boundary being submerged (for instance, a propeller of a ship or a turbine blade), the number of layers would ideally be twice as large still so that the ghost particles are influenced only by fluid particles from only one side of the fluid domain. Such cases could lead to an intractable number of particles rendering the simulation impossible to run on current hardware. A novel hybrid approach combining a fixed and mirrored method, where a fixed “standstill” is constructed dynamically for each particle can be found in reference [49]. While potentially providing the best of both worlds in the GPBC methods, this new approach still requires some further work before it can be generally applicable.

Many other novel wall boundary conditions have also been proposed throughout the years. A rather comprehensive list of references describing such attempts can be found in the grand challenges section in the SPHERIC web page [136].

As a conclusion of this review, a brief overview of the properties of the wall boundary conditions considered for implementation are given in Table 4.2. These include the RFBC [102], the dynamic boundary condition (DBC) [29], the analytical boundary condition [45] and the SABC [5, 46, 92], and the fixed GPBC [3, 90]. The boundary conditions in references [29, 102] can be discarded as they do not offer sufficient accuracy, and also the method in reference [45] because it is limited to two dimensions. Of the remaining methods, the fixed ghost particle method by Adami et al. [3] is the least complicated to implement, is general enough for the scope of this thesis, is very efficient, and it has also shown to be adequate for further development to multi-fluid simulations and surface tension. For all these reasons, the fixed ghost particle method was chosen for implementation, and it is described in more detail hereafter.

Table 3.1: Comparison of solid walls boundary condition features.

Name and reference	Kernel support	Physically accurate	Implementation	Comments
Repulsive force [102]	Incomplete	No	Simple	Requires a tuning parameter
Dynamic [29]	Incomplete	No	Extremely simple	Extremely efficient and flexible
Analytic γ [45]	Truncated	Yes	Complicated	Never was extended to 3D and very inefficient
Semi-analytic [46] [92]	Truncated	Yes	Very complicated	Analytic approach for γ , complicated in 3D
Semi-analytic [5]	Truncated	Yes	Quite complicated	Numerical approach for γ , losses accuracy
Fixed ghost particles [90]	Full	Yes	Quit complicated	Requires special treatment in corner regions
Fixed ghost particles [3]	Full	Yes	Quite simple	No special treatment in corners regions

3.4.2 Fixed ghost particle method with force balance correction

The main idea of this method is to prescribe fixed ghost particles with pressure values obtained by a force balance on either side of a solid wall to ensure that no particle penetrates the solid walls and that the fluid pressure field stays smooth. A discussion on how to assign the ghost particles and their required values of pressure, density and velocity follows. The description is limited to the aspects of this method that are relevant to this work. Full details of the method are given in reference [3].

In this method a number of layers, N , of fixed ghost particles are placed on the outer side of a solid boundary to assure that fluid particles always have a *full kernel support*. According to equation (3.31) this translates to $N = \Delta x / (w_h h)$ where w_h is the coefficient of the kernel support, h is the smoothing length, and Δx is the initial particle spacing. For example, when using a kernel with $w_h = 2$ the smoothing length can be chosen to be $h = 1.5\Delta x$ so that $N = 3$. Figure 3.6 illustrates the use of fixed ghost particles at the wall and the notation used.

To obtain a continuous flow field, the ghost particles are assigned pressure values obtained by a force balance on either side of the solid wall as

$$p_w = \frac{\sum_f p_f W_{wf} + \sum_f \rho_f (\mathbf{g} + \mathbf{a}_w) \cdot (\mathbf{x}_w - \mathbf{x}_f) W_{wf}}{\sum_f W_{wf}} \quad (3.57)$$

where the subscripts f and w denote the fluid and the ghost particles, respectively, and $\mathbf{a}_w$ is the prescribed wall acceleration. The density of the ghost particles is

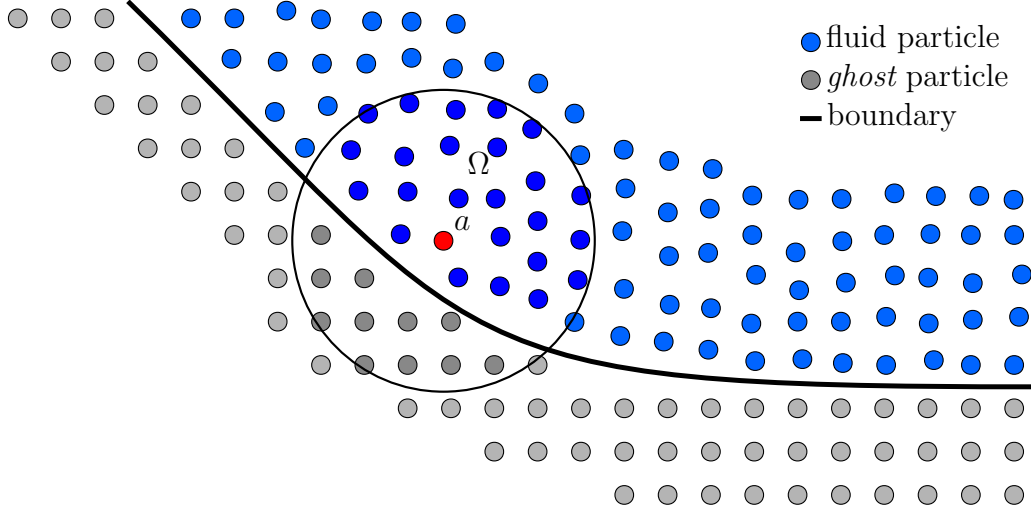


Figure 3.6: Illustration of the fixed ghost particles boundary condition of Adami et al. [3] with the notation used.

obtained from the equation of state (3.20) as

$$\rho_w = \rho_0 \left(\frac{p_w}{p_0} + 1 \right)^{\frac{1}{\gamma}} \quad (3.58)$$

with $p_0 = \rho_0 c_0^2 / \gamma$. This treatment ensures that there is no fluid particle penetration of the solid wall and that the fluid pressure field adjacent to the solid walls is smooth. The ghost particles have a constant mass, taken to be equal to the initial mass of a fluid particle and their velocity is calculated as follows. The velocity is set to the specified wall velocity, $\mathbf{v}_a$, in the continuity equation (3.36), while in the momentum equation (3.37) the velocity is treated depending on the wall slip condition.

To approximate a *free-slip* condition of the form $\mathbf{v} \cdot \mathbf{n} = 0$, the interaction of a fluid particle with a ghost particle ignores the viscous term and the ghost particle velocities do not have to be calculated.

To impose a *no-slip* condition of the form $\mathbf{v} = \mathbf{v}_s$, where $\mathbf{v}_s$ is the specified wall velocity (i.e. $\mathbf{v}_s = 0$ when the wall is fixed), the velocity field is first extrapolated onto the ghost particle's positions by

$$\tilde{\mathbf{v}}_a = \frac{\sum_b \mathbf{v}_b W_{ab}}{\sum_b W_{ab}} \quad (3.59)$$

and the ghost particle's velocity is

$$\mathbf{v}_w = 2 \mathbf{v}_s - \tilde{\mathbf{v}}_a. \quad (3.60)$$

3.4.3 Fixed ghost particle boundary condition errors

In instances when very few isolated fluid particles are moving very slowly in parallel to or away from the boundary, such as after a wave impacting a wall, the fluid particles can get stuck on the boundary or even penetrate it. This often affects only a very limited number of particles and has no measurable effect on the bulk of the fluid. In fact, these particles are commonly removed either during the simulation (if their density values go over or under predefined limits or entirely leave the domain) or during post-processing where they are simply neglected.

While the exact mechanism responsible for this error is yet unknown, it is assumed to be related to the fact that the values of isolated fluid particles are very poorly reproduced without enough information from neighbouring particles due to the kernel approximation. These inaccurate values are then used to interpolate the corresponding *ghost* particles values and the fluid boundary particle interaction. A possible hypothesis is that the force applied by the boundary particles on a fluid particle acts in the opposite direction to the fluid velocity. When a particle is moving towards the wall, the force applied by the boundary pushes it away, as expected, but if the fluid particle is moving away from the solid wall, a small attraction force is generated which slowly pulls the fluid particle towards it. In the case where a fluid particle is surrounded by other fluid particles, the inter-particle forces between them overcome this attraction between the boundary and the fluid. However, when the fluid particle is isolated or has a small number of neighbours, the attraction force overcomes the inter-particle fluid forces and pulls the fluid particle *towards* and *in to* the wall. This issue is pervasive in boundary condition formulations and should be properly investigated and addressed.

A possible solution to this problem could be to create a thin “buffer zone” adjacent to the boundary in which for fluid particles moving away from the boundary, i.e. $(\mathbf{v}_f - \mathbf{v}_w) \cdot (\mathbf{x}_f - \mathbf{x}_w) > 0$, the fluid-boundary particle interaction is neglected. The applicability of this approach is also left for future investigation.

A further issue specific to the fixed ghost particle wall boundary condition of Adami et al. [3] is the interpolation for the pressure values of the ghost particles in equation (3.57). As the ghost particles interact only with the fluid particles, the kernel support in the interpolation procedure is not full leading to inaccurate kernel approximations. To address this, the pressure interpolation is divided by the sum of the kernel, which is equivalent to a zeroth-order re-normalisation of the kernel. Linear and higher order errors are thus expected and are noticed by the slight variations in

the pressure field adjacent to the wall.

It is important to emphasise that despite these numerical issues, this treatment of the solid walls still offers one of the most accurate representations found in the literature.

3.5 Dissipation models

A common feature in the SPH schemes is the need to add dissipation to obtain meaningful simulations. Like most CFD methods, SPH has two types of dissipation: *physical* dissipation which is used to model viscous effects, and *numerical* dissipation which is used to stabilise the scheme and overcome spurious numerical effects generated by the initial and boundary conditions or otherwise.

Various dissipative terms have been proposed in the SPH literature. The most common term is the artificial viscosity [98], based on the Neumann-Richtmyer artificial viscosity, which provides numerical dissipation in the momentum equation (3.37). These terms are effective in reducing oscillations appearing from pseudo-sound waves generated by the fluid particles not being initially in a zero-energy state [115] and helps stabilise the scheme. This term is also commonly used to simulate physical viscosity [65], due to its similarity to the physical viscosity models, despite being problem dependent and considered by some practitioners to be unsuitable.

Similarly, there are several laminar viscosity terms available in the SPH literature, but their applicability to real world engineering problems (where it is common to use artificial viscosity instead) has not thoroughly been examined. A few such schemes which seem promising based on the literature are described in references [3, 84, 104] and in the following. Turbulence modelling is still actively researched in SPH, see for instance [46], but this is outside the scope of this thesis.

Another numerical artefact in the SPH scheme is the appearance of spatial fluctuations in the density (and hence also pressure) field. The problem arises from various sources. The first is the inconsistency between the continuity equation (3.36) which in WCSPH does not exactly conserve mass [105] and the fixed mass value m_b which does explicitly conserve mass [23]. Another is the emergence of spurious modes from the evaluation of function derivatives at the same location as the variables. This problem is known as the *chequerboarding problem* in finite difference methods [123] and was shown to occur also in SPH [11, 44]. An additional source is the use of a stiff equation of state (3.20) which does not account for thermal effects [147]. To address

these issues, several numerical dissipation schemes applied to the density field have been proposed [8, 23, 44, 47, 93, 96] which are discussed in Section 3.5.3.

3.5.1 Artificial viscosity

To damp or smooth spurious numerical oscillations, a dissipative term, $\mathbf{\Pi}$, is added to the momentum equation (3.18) that mimics a physical viscosity (and hence it is referred to as artificial viscosity), leading to

$$\frac{D\mathbf{v}_a}{Dt} = - \sum_b m_b \left(\frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} + \mathbf{\Pi}_{ab} \right) \nabla_a W_{ab} + \mathbf{f}_e. \quad (3.61)$$

Several formulations of the artificial viscosity term exist, but the most commonly used is that of Monaghan [98], given by

$$\mathbf{\Pi}_{ab} = \begin{cases} -\alpha \frac{h \bar{c}_{ab} \mu_{ab}}{\bar{\rho}_{ab}}, & \text{if } (\mathbf{v}_a - \mathbf{v}_b) \cdot (\mathbf{x}_a - \mathbf{x}_b) < 0 \\ 0, & \text{otherwise} \end{cases}; \quad \mu_{ab} = \frac{(\mathbf{v}_a - \mathbf{v}_b) \cdot (\mathbf{x}_a - \mathbf{x}_b)}{(\mathbf{x}_a - \mathbf{x}_b)^2 + \epsilon^2} \quad (3.62)$$

where α is a problem-dependent tuning parameter and quantities with an over bar refer to average between particles calculated as $\bar{\psi}_{ab} = \frac{1}{2}(\psi_a + \psi_b)$. ϵ is a parameter added to ensure a non-zero denominator when particles are in the same position or on top of each other, commonly set as $\epsilon^2 = 0.01h^2$. In practice, it was found that this parameter is unneeded and hence removed in the implemented code. According to [65], in the presence of physical viscosity, α can be related to an effective kinematic viscosity ν as

$$\nu = \frac{\alpha h \bar{c}_{ab}}{2(d+2)} \quad (3.63)$$

where d is the dimension of the simulation. This is often the relation used literature when a “numerical” Reynolds number is discussed in relation to the artificial viscosity term.

3.5.2 Laminar Viscosity

In the laminar viscosity models the momentum equation is written as

$$\frac{D\mathbf{v}_a}{Dt} = - \sum_b m_b \left(\frac{\rho_b^{2k} P_a + \rho_a^{2k} P_b}{(\rho_a \rho_b)^{k+1}} \right) \nabla_a W_{ab} + \mathbf{\Pi}_a + \mathbf{f}_e, \quad (3.64)$$

where $\mathbf{\Pi}_a$ is the viscous stress term.

Various formulations of $\mathbf{\Pi}_a$ have been proposed in the literature. One such a term was proposed by Lo and Shao [84] and reads

$$\mathbf{\Pi}_a = \sum_b 4m_b \frac{\nu(\mathbf{x}_a - \mathbf{x}_b) \cdot \nabla_a W_{ab}}{(\rho_a + \rho_b)(\mathbf{x}_a - \mathbf{x}_b)^2} (\mathbf{v}_a - \mathbf{v}_b) \quad (3.65)$$

where ν is the kinematic viscosity of the fluid.

Alternatively, a similar term was used by Adami et al. [3] which reads

$$\mathbf{\Pi}_a = \frac{1}{m_a} \sum_b \frac{2\eta_a\eta_b}{\eta_a + \eta_b} (V_a^2 + V_b^2) \frac{(\mathbf{x}_a - \mathbf{x}_b) \cdot \nabla_a W_{ab}}{(\mathbf{x}_a - \mathbf{x}_b)^2} (\mathbf{v}_a - \mathbf{v}_b) \quad (3.66)$$

where η_a is the dynamic viscosity of particle a . This formulation is very similar to (3.65) but is reported to work better for multiple fluids because it uses the square of the volume divided by a combination of the particle masses, rather than the average of the density, and takes the averages of the viscous terms. It is noted that in the literature the ϵ variable used to keep the denominator non-zero in equation (3.62) is sometimes included in both these models as well. However, it was found to be redundant in practice and hence was not included here.

In the laminar viscosity formulations above the second order derivative term has the dot product between the difference in positions and gradient of the kernel, while in the artificial viscosity formulation (3.62) the dot product is between the positions and the velocities. The former better approximates a Laplacian of the velocity [146] and models viscous forces more accurately [105], however, unlike the later, it dose not strictly conserves angular momentum. For this reason, and despite its reduced accuracy, some SPH practitioner prefer to use artificial viscosity with the relation in equation (3.63) to mimic laminar viscosity, especially in cases involving large fluid velocities [105].

3.5.3 Density dissipation

A variety of methods to address the spurious density fluctuations in SPH have been proposed over the years. Colagrossi and Landrini [23] suggested applying a filter every number of time-steps to the density field to smooth these density fluctuations.

This filtering is either performed with a zeroth-order *Shepard filter* given by

$$\rho_a^{new} = \frac{\sum_b m_b W_{ab}}{\sum_b (m_b / \rho_b) W_{ab}} \quad (3.67)$$

or with a first-order moving least-squares (MLS) filter [23]. However this procedure does not conserve volume and leads to significant errors in long duration simulations [7].

More recently it was suggested to include numerical dissipation to the continuity equation (3.36), commonly known as the δ -SPH scheme [96]. This model modifies the continuity equation to read

$$\frac{D\rho_a}{Dt} = \frac{1}{\rho_a^{2k-2}} \sum_b m_b \left((\rho_a \rho_b)^{k-1} (\mathbf{v}_a - \mathbf{v}_b) \right) \cdot \nabla_a W_{ab} + \mathcal{D}_a \quad (3.68)$$

and

$$\mathcal{D}_a = 2\delta h c_0 \sum_b V_b \psi_{ba} \frac{(\mathbf{x}_b - \mathbf{x}_a) \cdot \nabla_a W_{ab}}{(\mathbf{x}_b - \mathbf{x}_a)^2}. \quad (3.69)$$

Here, δ is a tuning parameter scaled by the smoothing length h and the numerical speed of sound c_0 to ensure dimensional consistency, and k is an arbitrary parameter as in equation (3.36). $\mathcal{D}_a$ is a dissipative term which approximates a Laplacian [145] of ψ_{ba} where ψ_{ba} can take various forms. The second-order term proposed in reference [96] reads

$$\psi_{ba} = \rho_b - \rho_a. \quad (3.70)$$

Ferrari et al. [47] proposed a similar second-order dissipative term based on Riemann solvers using a Rusanov flux. It has no tuning parameter and is capable of stabilising SPH without the addition of the classical artificial viscosity in inviscid flows.

Both these schemes provide significant improvements in the quality of simulations. However, like the filtering procedure, they do not conserve volume, are incompatible with the hydrostatic solution, and dissipate too much energy [7]. Volume conservation is of great importance, particularly for long duration simulations. Lack of volume conservation gradually leads to a growing particle spacing towards the free surface rendering the simulation unusable. This behaviour has been confirmed and it is shown in Section 5.1.3. Despite their shortcomings, the increase in volume remains negligible over short periods of time [7] and these methods are commonly used due to their effectiveness in dissipating the spurious pressure oscillations and their simplicity.

Antuono et al. [8] proposed a correction to the δ -SPH scheme [96] that employs a fourth-order conservative dissipation term

$$\psi_{ba} = (\rho_b - \rho_a) - \frac{1}{2} \left(\tilde{\nabla} \rho_b + \tilde{\nabla} \rho_a \right) \cdot (\mathbf{x}_b - \mathbf{x}_a) \quad (3.71)$$

where $\tilde{\nabla} \rho_a$ is the corrected gradient of ρ_a . This correction is obtained by a first-order re-normalisation to the approximation of the kernel derivative using the moving least-squares (MLS) method [13] to ensure global convergence [8]. It is calculated as

$$\tilde{\nabla} \rho_a = \sum_b (\rho_b - \rho_a) \mathbf{L}_a \nabla_a W_{ab} \quad (3.72)$$

with the MLS correction matrix

$$\mathbf{L}_a = \left[\sum_b (\mathbf{x}_b - \mathbf{x}_a) \otimes \nabla_a W_{ab} \right]^{-1}. \quad (3.73)$$

where $\otimes$ denotes the vector cross product.

This term dissipates the spatial pressure oscillations without causing non-physical effects on the fluid. It conserves volume, dissipates less energy, and stays stable over long periods of time [7, 8]. It also has a stabilising effect that permits the removal of the classical artificial viscosity term from the momentum equation in inviscid flows. Both terms can, and often are, used in tandem. The main drawback of the method is that it is significantly more expensive than the second-order method. The implementation requires two additional loops through the fluid particles and the solution of a 2×2 (in two dimensions) or 3×3 (in three dimensions) linear system of equations for each particle. For these reasons, and despite its drawbacks, this scheme has been assessed to be the most suitable for the work in this thesis.

A couple of other notable schemes which address this issue are the following. Fatehi and Manzari [44] address the *chequerboarding problem* by adding a dissipative term to the continuity equation based on the difference between two different discretisation of the pressure gradient, similar to schemes deployed in collocated grid strategies for incompressible flows. Mayrhofer et al. [93] proposed a similar second-order scheme to those proposed in references [47, 96] with an additional term to account for the hydrostatic solution. These methods are not included in this work and their investigation in the context of long duration sloshing is left for future study.

3.5.4 Damped hydrostatic initialisation

The SPH standard initialisation uses particles arranged on a Cartesian grid with values set to the hydrostatic solution. However, this set-up infers a non-zero-energy initial condition that creates pseudo-sound waves which travel through the fluid domain for a period of time until dissipated by various means (such as numerical or physical dissipation) [115]. A method for reducing these waves is to initialise the fluid particles with a density set to the initial density (so that pressure is zero everywhere) and the external acceleration is set to zero, then gradually increase the acceleration to its desired value, for instance that of gravity. During this period, the fluid particles rearrange themselves to the hydrostatic solution naturally. An example of this approach can be found in reference [102] where the acceleration of each particle in the momentum equation (3.18) is damped by a factor ζ given by

$$\zeta(t) = 0.5 \left[\sin \left(\left(-0.5 + \frac{t}{t_d} \right) \pi \right) + 1 \right] ; \quad t \leq t_d \quad (3.74)$$

where t_d is the damping period. This damping period allows for particles to reorganise in to a lower energy state thus reducing the initial pressure waves. However, this approach does not fully remove the initial waves, increases the simulation time by the damping period and can be difficult to implement in cases where the fluid does not start from rest or with a hydrostatic initialisation (such as dam breaks or water jets). This method can be useful for short duration simulations which start from a hydrostatic solution, but has negligible effects over long periods as the numerical dissipation schemes eventually overcome the initial waves. For these reason the method was not used in this thesis other than where explicitly mentioned.

3.6 Governing equations in a moving frame of reference

To simulate cases when the domain is subjected to external excitation within the SPH framework, such as sloshing, it is more useful to account for a moving frame of reference rather than devising moving boundaries conditions.

Following the derivation given in reference [43], the momentum equation (3.1) in a moving frame of reference can be re-written as

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho} \nabla \sigma + \mathbf{g} - \mathbf{a}_w - \omega \times \mathbf{v}_w - 2\omega \times \mathbf{v} - \dot{\omega} \times \mathbf{x} - \omega \times (\omega \times \mathbf{x}) \quad (3.75)$$

where $\mathbf{g}$ represents a constant force such as gravity, the subscript w represents the frame variables and ω denotes the angular velocity.

In essence, the only change in the SPH formulation is the addition of extra terms to the external force $\mathbf{f}_e$ of the the momentum equation (3.18), which is now given as

$$\mathbf{f}_e = \mathbf{g} - \mathbf{a}_w - \omega \times \mathbf{v}_w - 2\omega \times \mathbf{v}_a - \dot{\omega} \times \mathbf{x}_a - \omega \times (\omega \times \mathbf{x}_a). \quad (3.76)$$

For purely lateral excitation, the instantaneous angular momentum is zero and the result is a greatly simplified expression with only the addition of the external forcing term

$$\mathbf{f}_e = \mathbf{g} - \mathbf{a}_w. \quad (3.77)$$

3.7 Time integration

The initial-value problem (3.17 – 3.19) is integrated in time using a suitable solver for systems of ordinary differential equations. It is generally accepted that geometric-integration algorithms are best suited for the solution of the SPH equations as they preserve the Hamiltonian structure of the system [14].

A classical method widely utilised in molecular dynamics simulations and SPH is the Verlet scheme. Using the abridged notation $\mathbf{U}_a = [\rho_a, \mathbf{v}_a]^T$ and $\mathbf{R}_a = [M_a, \mathbf{F}_a]^T$ in the semi-discrete form of the SPH equations (3.17 – 3.19) with the terms described in Section 3.1.6, the variables are integrated in time according to

$$\mathbf{U}_a^{n+1} = \mathbf{U}_a^{n-1} + 2\Delta t \mathbf{R}_a^n \quad (3.78)$$

and the position of the particles is updated according to

$$\mathbf{x}_a^{n+1} = \mathbf{x}_a^n + \Delta t \mathbf{v}_a^n + \frac{\Delta t^2}{2} \mathbf{F}_a^n \quad (3.79)$$

that results from a Taylor series expansion, and the pressure is updated at the end of the step by using the equation of state (3.20). However, the conservation of mass and momentum equations(3.78) are decoupled from the equation for updating of the position (3.79) which can lead to the scheme diverging. In order to avoid divergence,

an Euler scheme is employed every N_s time steps, given as

$$\mathbf{U}_a^{n+1} = \mathbf{U}_a^n + \Delta t \mathbf{R}_a^n \quad (3.80)$$

$$\mathbf{x}_a^{n+1} = \mathbf{x}_a^n + \Delta t \mathbf{v}_a^n + 0.5 \Delta t^2 \mathbf{F}_a^n. \quad (3.81)$$

A typical value is to use $N_s = 50$ [28], however values as small as $N_s = 10$ are could be required.

An alternative time stepping scheme which also preserves the Hamiltonian is the symplectic predictor-corrector scheme [51], that reads

$$\begin{aligned} \rho_a^{n+\frac{1}{2}} &= \rho_a^n + \frac{\Delta t}{2} M_a^n \\ \mathbf{v}_a^{n+\frac{1}{2}} &= \mathbf{v}_a^n + \frac{\Delta t}{2} \mathbf{F}_a^n \\ \mathbf{x}_a^{n+\frac{1}{2}} &= \mathbf{x}_a^n + \frac{\Delta t}{2} \mathbf{v}_a^n \end{aligned} \quad (3.82)$$

$$\begin{aligned} \rho_a^{n+1} &= \rho_a^n \left(\frac{2 - \epsilon}{2 + \epsilon} \right); \quad \epsilon = -\Delta t \left(\frac{M_a}{\rho_a} \right)^{n+\frac{1}{2}} \\ \mathbf{v}_a^{n+1} &= \mathbf{v}_a^n + \Delta t \mathbf{F}_a^{n+\frac{1}{2}} \\ \mathbf{x}_a^{n+1} &= \mathbf{x}_a^n + \frac{\Delta t}{2} (\mathbf{v}_a^n + \mathbf{v}_a^{n+1}). \end{aligned}$$

Here the pressure P_a is calculated with the equation of state (3.20) at each half-step. This scheme proved to be more stable than the Verlet scheme for the simulations conducted in this thesis, and thus it was exclusively used.

Both these schemes are explicit, simple to program and reasonably efficient for advection-dominated flows. However, due to their explicit nature they are subject to time-step stability restrictions that depend on the advection, diffusion and forcing terms. Following [103] a particle's time-step, Δt_a , is calculated as

$$\Delta t_a = C_{\Delta t} \min(\Delta t_f, \Delta t_{cv}); \quad \begin{cases} \Delta t_f = \min_b \sqrt{\frac{h}{\|\mathbf{F}_{ab}\|}} \\ \Delta t_{cv} = \min_a \frac{h}{c_0 + \max_b \left| \frac{h(\mathbf{v}_a - \mathbf{v}_b) \cdot (\mathbf{x}_a - \mathbf{x}_b)}{\|\mathbf{x}_a - \mathbf{x}_b\|^2} \right|} \end{cases} \quad (3.83)$$

and $\Delta t = \min_a \Delta t_a$. The time-step Δt_f is based on the value for force per unit mass $\|\mathbf{F}_{ab}\|$, as calculated in (3.18), and is used to avoid particles moving too close to their neighbours or penetrating the walls [97]. The value Δt_{cv} arises from a Courant-

Friedrichs-Lewy (CFL) like time-step restriction which combines the classical CFL condition based on the numerical speed of sound with the viscous time-step restriction [97]. The tuning constant, $C_{\Delta t}$, known as the Courant number, is normally taken to be $0.1 \leq C_{\Delta t} \leq 0.3$. Here, $C_{\Delta t} = 0.1$ has been used.

3.8 Post-processing

The Lagrangian nature of SPH poses different challenges to the post-processing of the simulated results to those posed by traditional grid-based Eulerian CFD methods. SPH particles carry information with them, so that the resolution follows the flow. This means that visualising the fluid domain is as simple as plotting the particles and colouring them according to the desired properties. However, when conducting experiments variables are usually measured at specified fixed locations. In SPH simulations there is no guarantee of any fluid particles being in the exact locations of interest at any given time, making it impossible to directly take a measurement there. To achieve measurements at a fixed position, an SPH interpolation, using equation (3.6), has to be performed around the point(s) of interest. The interpolation carries all the associated issues of a SPH approximation, such as a loss of accuracy due to the numerical errors of the method and the additional computational cost.

As an example, Figure 3.7(a) shows a snapshot of a SPH simulation of a dam break which is compared to an experiment, taken from Section 5.3. In the experiment, a pressure gauge, labelled P1, is placed on the far wall. In order to verify the simulation, pressure values at the location of the gauge needs to be compared with those of the experiment. The enlarged views in Figure 3.7(a) show that no fluid particles are present exactly at the location of the pressure gauge. The location of the gauge is used as an interpolation point where the pressure is evaluated at given time intervals in order to obtain the pressure time history which is compared against the experimental values as seen in Figure 3.7(b).

The following sections will describe how to calculate the fluid height at specific locations or capture the free surface, the total energy or energy components in a purely post-processed manner, and discuss how to best obtain fluid forces.

3.8.1 Free-surface detection and height measurements

A very attractive feature of the SPH method is that there is no need to explicitly track the free surface of SPH as explained in Section 3.3. However, it is often very

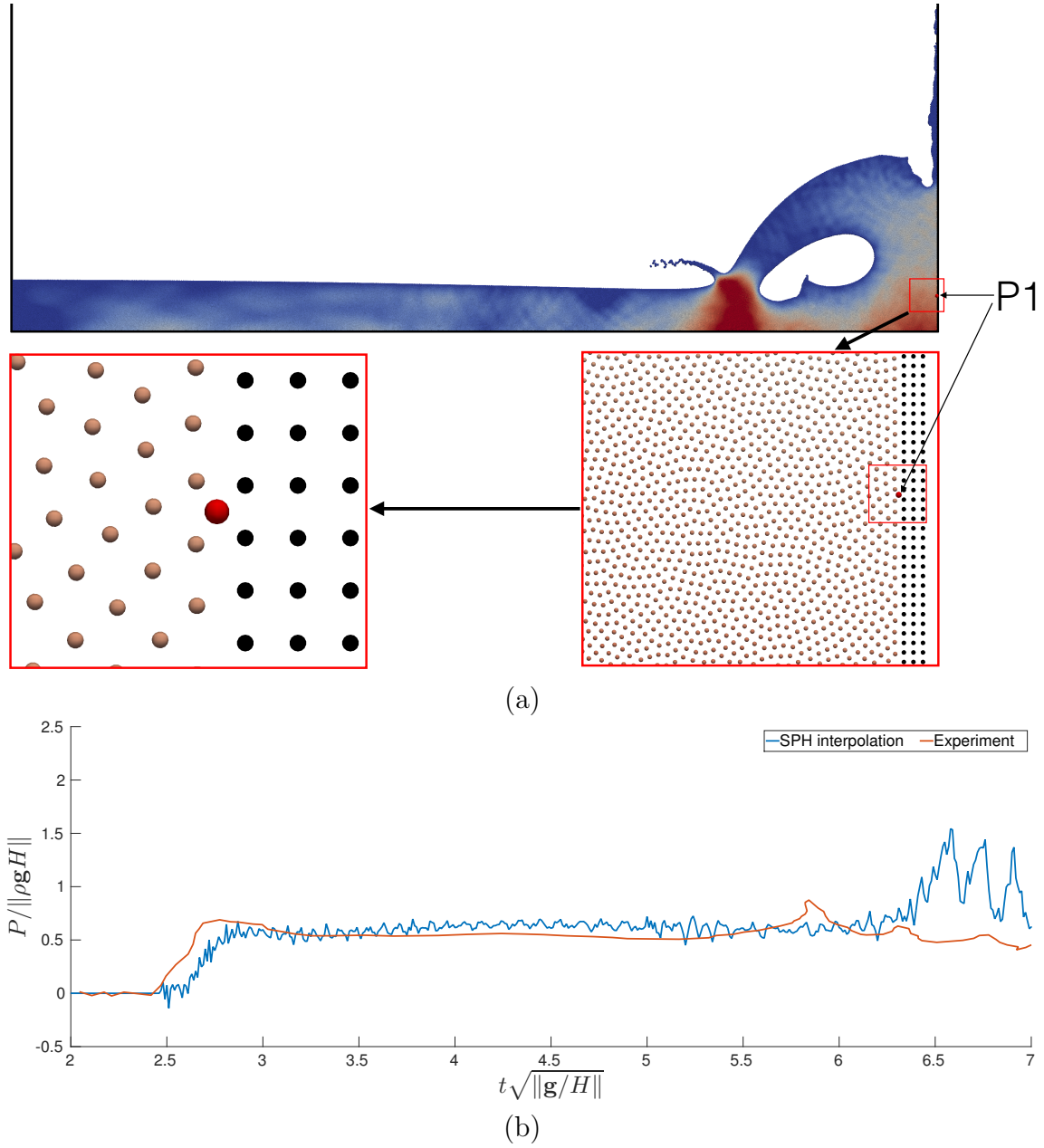


Figure 3.7: A pressure interpolation post processing example of a comparison between an SPH simulation and an experiment of a dam break: (a) snapshot from the SPH simulation showing the location of a pressure gauge, P1, at different levels of magnification; (b) comparison of the pressure time history at P1 between the interpolated values obtained from the SPH simulation and the experimental data.

useful to extract the free surface information for either computational or visualisation purposes. Marrone et al. [91] propose a method which is very accurate and also includes surface normal information, but it requires extra computations during the SPH runtime, is complicated to implement and is computationally demanding. Therefore, an alternative method has been developed which is far cheaper and faster to compute, albeit less accurate, and can be employed as a post-processing tool with no additions to the SPH scheme. This method simply interpolates the mass value of each particle and identifies particles with mass smaller than a prescribed value. Usually, the values used are $0.4m_f$ in two dimensions and $0.5m_f$ in three dimensions, where m_f is the mass of the fluid. In this way, any interpolation point with a value close to but lower than this threshold is identified as being on the free surface. The interpolated mass value is lower for free-surface particles because the kernel support is not full at the free surface (see Figure 3.3) and the SPH interpolation in equation (3.6) is lower by a factor of the missing kernel support. As an example, the free surface of the snapshot given in Figure 3.7 is shown in Figure 3.8.

This same approach is available in the DualSPHysics post-processing tools to measure the maximum height of the fluid at specific locations. By interpolating the mass at a given (x, y) position and taking the maximum of all z particles which are identified to be on the free-surface, the maximum fluid height at a given position can be found. This technique is used in the remainder of the thesis as the means to measure the wave heights.

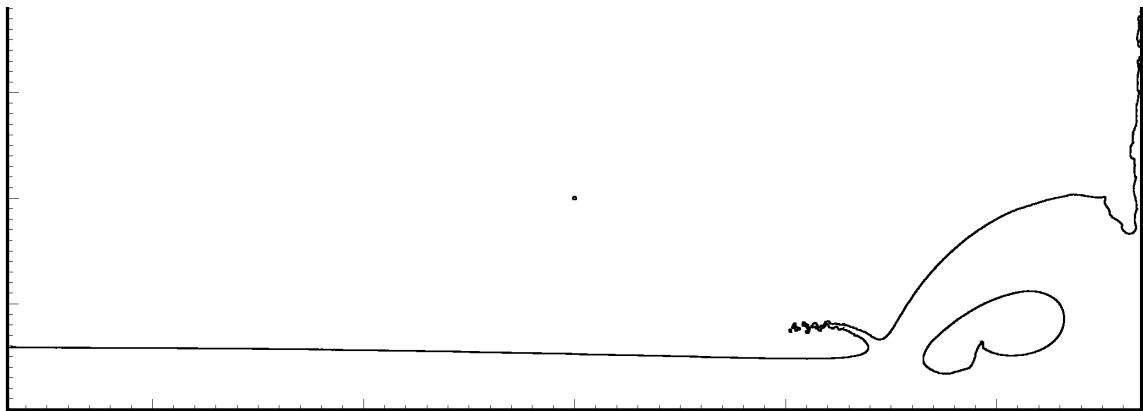


Figure 3.8: The free surface plot of a SPH simulation of a dam break case. Taken at the same instance as Figure 3.7 (a).

3.8.2 Energy calculation

Given the particle nature of the SPH method, the energy of the system can be derived from the Hamiltonian of the set of particles [146]. The total energy offers a unique way of assessing the quality of a SPH scheme by measuring its energy dissipation. For instance, in a still water case an *ideal* scheme should perfectly conserve energy. However due to various choices of formulation, boundary conditions and numerical dissipation schemes, this is rarely the case. Observing the total energy deviation from the ideal static case allows for a qualitative assessment of these choices and provides a rigorous methodology for selecting the best ones.

The Hamiltonian $\mathcal{H}$, a measure of the energy of a system, is given as the sum of kinetic and potential energy

$$\mathcal{H} = E_{\text{kin}} + E_{\text{pot}} \quad (3.84)$$

which for an isolated system in the absence of dissipation due to viscous stresses, is conserved.

The kinetic energy for a set of N_p discrete particles is given as

$$E_{\text{kin}} = \sum_{a=1}^{N_p} m_a \frac{v_a^2}{2}. \quad (3.85)$$

where v_a^2 is the dot product of the velocity, i.e. $\mathbf{v}_a \cdot \mathbf{v}_a$. The potential energy can be further decomposed into its internal E_{int} and external E_{ext} , components, where the internal energy depends on the interactions between the discrete particles, and the external energy depends on the interactions with an external system such as gravitational pull. The external potential energy due to gravity is

$$E_{\text{ext}}^g = \sum_{a=1}^{N_p} m_a \mathbf{g} \cdot \mathbf{x}. \quad (3.86)$$

The internal potential energy can be derived from the first law of thermodynamics

$$\delta E_{\text{int}} = \delta Q + \delta W \quad (3.87)$$

where δE_{int} is the change in the internal energy of the system, δQ is the heat transfer into the system, and δW is the work done on the system. Energy conservation equations and thermal effects are not included in the framework, so for the purposes of this work, δQ is dropped. The work done *on* the system is given by the change in

the volume of the system, referred to as the pressure-volume work, and is given by $\delta W = -PdV$, where P is the pressure of the system and dV the change in volume, so that

$$\delta E_{\text{int}} = -PdV. \quad (3.88)$$

From the continuity equation and taking the volume as $V = m/\rho$, the change in the internal energy can be obtained as

$$\delta E_{\text{int}} = \frac{mP}{\rho^2} d\rho \quad (3.89)$$

where the pressure is calculated from the equation of state (3.20). The internal energy is then calculated by integrating δE_{int} leading to

$$E_{\text{int}}(\rho_f) = \int_{\rho_i}^{\rho_f} \frac{mP}{\rho^2} d\rho + E_{\text{int}}(\rho_i) = m p_0 \int_{\rho_i}^{\rho_f} \frac{1}{\rho^2} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right) d\rho + E_{\text{int}}(\rho_i). \quad (3.90)$$

Evaluating the integral from the initial density $\rho_i = \rho_0$ and the final density $\rho_f = \rho_t$, where t is the current time of the simulation, the internal energy reads

$$E_{\text{int}}(\rho) = \frac{m p_0}{\rho_0(\gamma - 1)} \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma-1} + \frac{\rho_0}{\rho}(\gamma - 1) - \gamma \right) + E_{\text{int}}(\rho_0). \quad (3.91)$$

Finally, for a system of N_p discrete particles the internal energy can be evaluated as

$$E_{\text{int}}(\rho) = \sum_{a=1}^{N_p} \frac{m_a p_0}{\rho_0(\gamma - 1)} \left(\left(\frac{\rho_a}{\rho_0} \right)^{\gamma-1} + \frac{\rho_0}{\rho_a}(\gamma - 1) - \gamma \right) + E_{\text{int}}(\rho_0). \quad (3.92)$$

The total energy of the system E_{tot} is then the sum of the kinetic energy E_{kin} (3.85), the external (potential) energy E_{ext}^g (3.86) and the internal energy E_{int} (3.92). For the purpose of post-processing, we are only interested in the conservation of energy and not its absolute value, giving the final expression for the total energy presented in the following section as

$$E_{\text{tot}}(t) = E_{\text{kin}}(t) + E_{\text{ext}}^g(t) + E_{\text{int}}(t) - (E_{\text{kin}}(t_0) + E_{\text{ext}}^g(t_0) + E_{\text{int}}(t_0)) \quad (3.93)$$

where t_0 is the initial time of the simulation.

For example, Figure 3.9 shows the evolution of the total energy in a still water tank using different features in the code taken from Section 5.1.2. It shows that the use of the dynamic boundary condition (DBC) [29] and the fixed ghost particle

boundary condition (GPBC) as described in Section 3.4.2 result in noticeable differences in the total energy dissipated. The full details and the analysis of this case are given in Section 5.1.2.

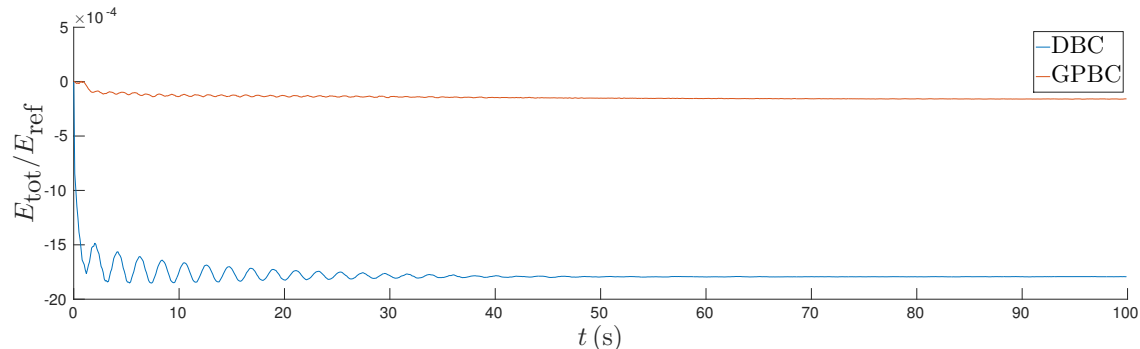


Figure 3.9: Normalised total energy over time of still water cases comparing the dynamic boundary conditions (DBC) and the fixed ghost particle boundary condition (GPBC). The total energy values are normalised by the potential energy at the start of the simulation.

Other methods for calculating the total energy of the system can be found in the SPH literature, for example [61]. However, they require the addition of the energy conservation equation to the scheme itself and cannot be performed solely at the post-processing stage.

3.8.3 Force calculations

Calculating the forces of the fluid or forces acting on the boundaries is often required in simulations, e.g. in fluid-structure interaction. Various methods can be used to obtain the forces in SPH simulation, and it is not intuitively clear which of these would be best.

A simple and naive approach to obtain fluid forces is to multiply the total acceleration of the particles by the mass to give the total force, i.e. $\mathbf{F} = \sum_a \frac{D\mathbf{v}_a}{Dt} m_a$. However, further investigation shows significant shortcomings of such a simple method, where the presence of oscillations in the pressure field, and hence the acceleration field, destroys the accuracy of the calculated forces.

A second approach is to integrate (interpolated) pressure values over the mean wetted surface of the domain, S_w , so that the force is

$$\mathbf{F} = \int_{S_w} P ds. \quad (3.94)$$

Placing the interpolation points for the pressure at a distance of $0.5h$ from the wall as to avoid spikes in the data was found to be the optimal distance when the fixed ghost particle (Section 3.4.2) method is employed – however the optimal distance might change based on the boundary condition used. This approach is very simple and efficient to implement in simple geometries in two dimensions, but it can become far more challenging when dealing with three dimensional domains.

A third and final option is to compute the force from the momentum conservation equation (3.37) for all or a subset of the boundary (or *ghost*) particles as

$$\mathbf{F} = \sum_w m_w \frac{D\mathbf{v}_w}{Dt} = \sum_w \sum_f m_w m_f \left(\frac{\rho_f^{2k} P_w + \rho_f^{2k} P_w}{(\rho_w \rho_f)^{k+1}} \right) \nabla_f W_{fw} + \mathbf{f}_e \quad (3.95)$$

where the subscript w denotes a boundary particle and f a fluid particle. Since the boundary particles only use the fluid particles for the calculation and do not include other boundary particles, the kernel support is truncated and this could lead to additional numerical errors. The computational costs of this method are larger than those of the alternatives.

To understand which of these methods produces the best results, a comparison of all three is performed. The force values obtained with each method are compared against those obtained analytically by potential flow theory for a linear sloshing case [43].

Figure 3.10(a) shows that the first method of calculating the force directly from the acceleration of fluid particles does not provide any meaningful results, and is completely discarded. The second and third methods show similar agreement with the analytical value, but a closer observation of their errors in Figure 3.10(b) shows that the third method, based on the acceleration of each boundary particle, has a significant advantage. The errors from the pressure integration method have a higher maximum value and tend to have irregular spikes, where as the errors obtained from the acceleration of the boundary particles show a far more regular pattern and grow in time as the simulation deviates from the analytical solution. This suggests that computing the forces through the acceleration of the boundary particles provides more consistent values and hence it is the method of choice for all subsequent force computations in the thesis.

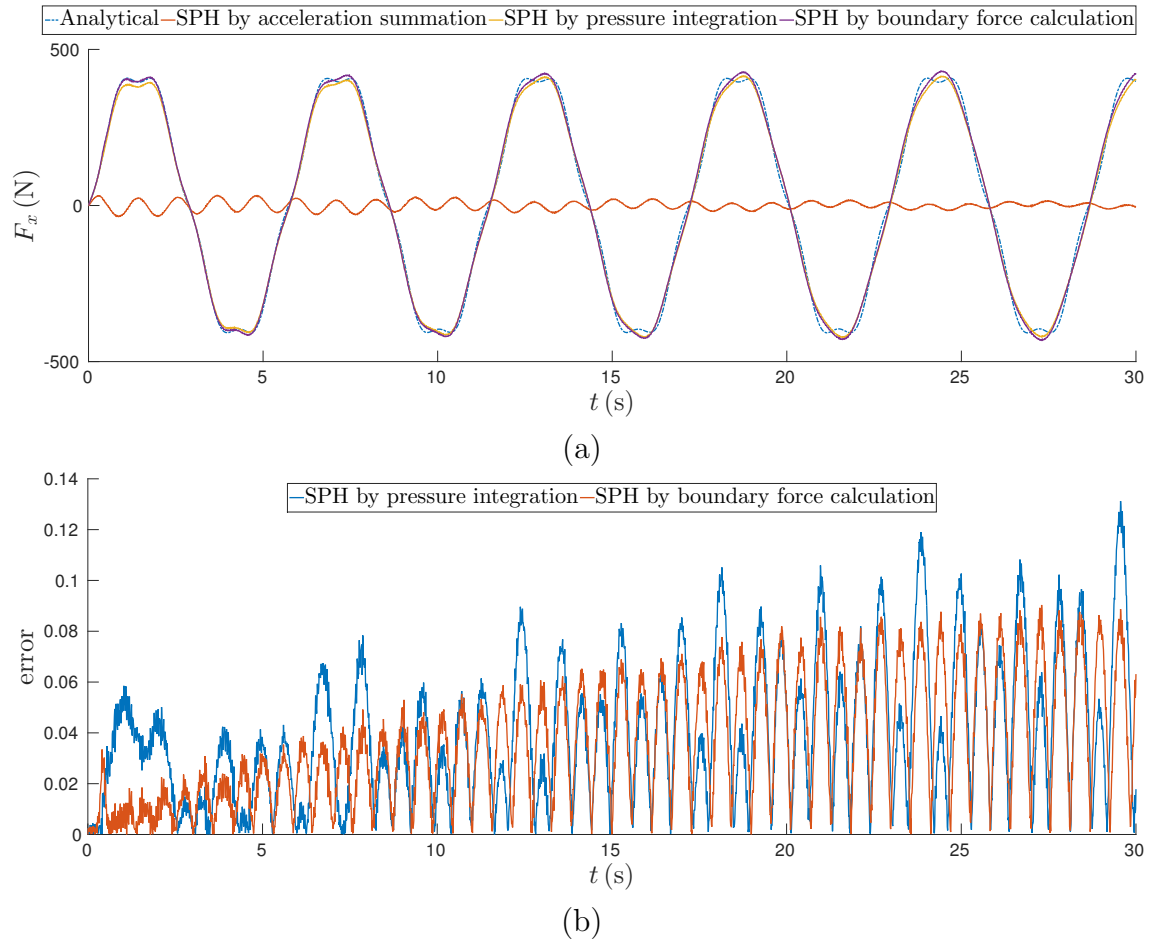


Figure 3.10: Comparison of different methods to compute fluid forces in SPH: (a) fluid force in x -axis direction (N) over time (s) comparing the three different methods of post-processing forces from the SPH simulation; (b) errors of the computed fluid forces between the pressure integral and the boundary acceleration post-processing methods. Errors are calculated as the absolute difference between the computed and analytical force, normalised by the maximum value of the analytical force.

Chapter 4

Parallel implementation on GPUs

In the last decade the single threaded central processing unit (CPU) power has reached its limit [139] and low energy consuming multi-core CPUs have become the standard. This had lead to the cost of high performance computing (HPC), i.e. computing servers containing many clusters of processing units all interconnected to deploy calculations on hundreds or even thousands of cores, to drop rapidly. As availability of multi-core CPUs and HPC clusters has become more widespread, a shift towards scalable massively parallel CFD codes which can efficiently run on very large numbers of cores is continuously growing. These massively parallel codes running on HPCs enable CFD practitioners to accurately simulate complicated problems which until recently were unattainable. The adoption of massive parallelisation can be seen in both commercial and academic CFD programs such as Flow3D [69], Ansys [6] and OpenFoam [25] amongst many others.

It was later realised that using the graphical processing unit (GPU), with its hundreds, and currently thousands, of *simple* processing cores, can be used for general purpose computing, a task traditionally done by the CPU. GPUs are optimised for fast logical or arithmetic (floating point) operations in order to render screen output in real time. The realisation that each of these parallel cores could be used to manipulate other types of data saw the start of general purpose GPU computing (GPGPU) [87]. The idea of a GPGPU is to use the GPU to offload specific types of calculations from the CPU, such as arithmetic and logical operations, greatly accelerating their processing time. In contrast to the CPU clusters in HPCs, GPUs offer massive parallelism on a single card which can run in a standard workstation with minimal costs, albeit with each parallel processor (core) far slower and less capable than for general tasks than a single CPU (core). Further, many GPUs

can be linked together in clusters to accelerate HPCs, replacing the role of the CPUs traditionally used for raw calculation, thus significantly increasing performance while reducing operational costs [94].

This chapter gives a brief introduction of the GPU architecture from both the hardware and software (programming) perspectives, covering the advantages and limitations compared to traditional CPU-based parallelisation. It is followed by an introduction to the open-source code DualSPHysics [28] and a discussion of why it was chosen for development in this work. Then it describes the modifications to the original code base, which include the implementation of the fixed ghost particle method and the corrected δ -SPH scheme as described in Sections 3.4.2 and 3.5.3 respectively. To conclude, the performance of the modified GPU code is analysed and compared to the original.

4.1 GPU architecture and programming

The GPU has large number of its transistors dedicated to arithmetic logic units (ALUs), in contrast to the CPU which dedicates a lot of transistors to performing data caching and control operations. Their architectures are compared in Figure 4.1. This gives the GPU a huge number of processors which specialise in computing

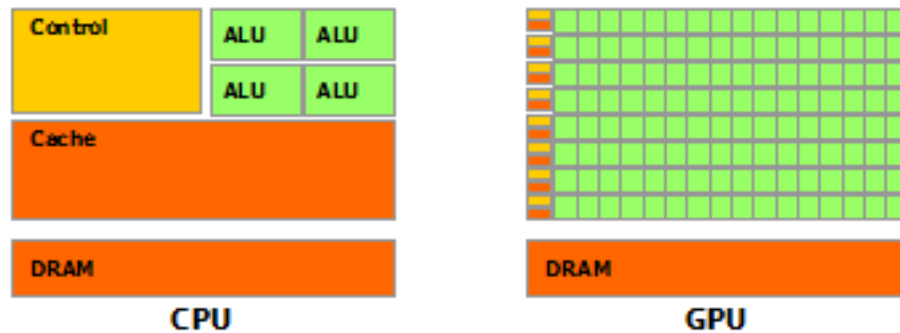


Figure 4.1: CPU vs GPU transistor allocation. Taken from reference [110].

the same instruction on many data elements in parallel, giving the GPU the ability to act, for a suitable problem, as a massively parallel hardware accelerator (also called a coprocessor). The role of a coprocessor is to offload a specific type of work which it is optimised to do (arithmetic and logical operations in this case) away from the CPU, thus accelerating their computational time. To instruct the GPU to

compute data, Nvidia developed the compute unified device architecture (CUDA) language [109] as an extension to C/C++, alongside a compiler called *nvcc*. The CUDA language is restricted to Nvidia hardware and is a propriety product. An alternative GPU programming languages is OpenCL (Open Computing Language [56]), which is a framework for programs that are executed across heterogeneous platforms including GPUs, CPUs and other processors. Other options are the use of API (Application Program Interface) compiler directives such as OpenACC [26] or Kokkos [40], which give directives in the code to offload specific loops from the CPU to a host coprocessor. Given that CUDA is the development language for DualSPHysics, the discussion in the following paragraphs is limited to the Nvidia architecture and to the CUDA language and terminology.

Both GPU and CPU contain a separate memory space, with the CPU in charge of transferring data to and from the GPU, and calling CUDA specific functions named kernel functions. A kernel function performs a set of instructions on each data element, with each data element occupying a CUDA thread. The threads are allocated into blocks, with each block containing up to 1024 threads on the latest hardware [111], which are executed on a streaming multiprocessor (SM). This way, the total number of threads to be processed is equal to the number of threads per block times the number of blocks [110]. The SM manages and executes threads in parallel groups of up to 32 threads, called a warp. To optimise performance, it is best to fill in the blocks in such a way that the largest amount of threads are computed per cycle. To achieve this, it is best to ensure that the threads per block are divisible by the warp size, which should be full, i.e. in multiples of 32 threads (for example 256 threads, or 8 warps per block). Blocks are then organised into a one-dimensional, two-dimensional, or three-dimensional grid of thread blocks (Figure 4.2(a)). The number of thread blocks in a grid is usually dictated by the size of the data being processed or the number of processors in the system, which can be much lower than the number of thread blocks.

GPU memory is managed in the following hierarchy. Each thread has a local memory and each block has a shared memory, while all threads (from all blocks) have access to the GPU global memory. The global memory is very large (several GB in top end GPUs) but also relatively slow (two orders of magnitude slower than local memory), while local memory is very fast but is only the size of a single thread. Shared memory can be as fast as local memory, but is limited to 64 KB on the latest Nvidia GPU architecture and is only used to transfer and share data across

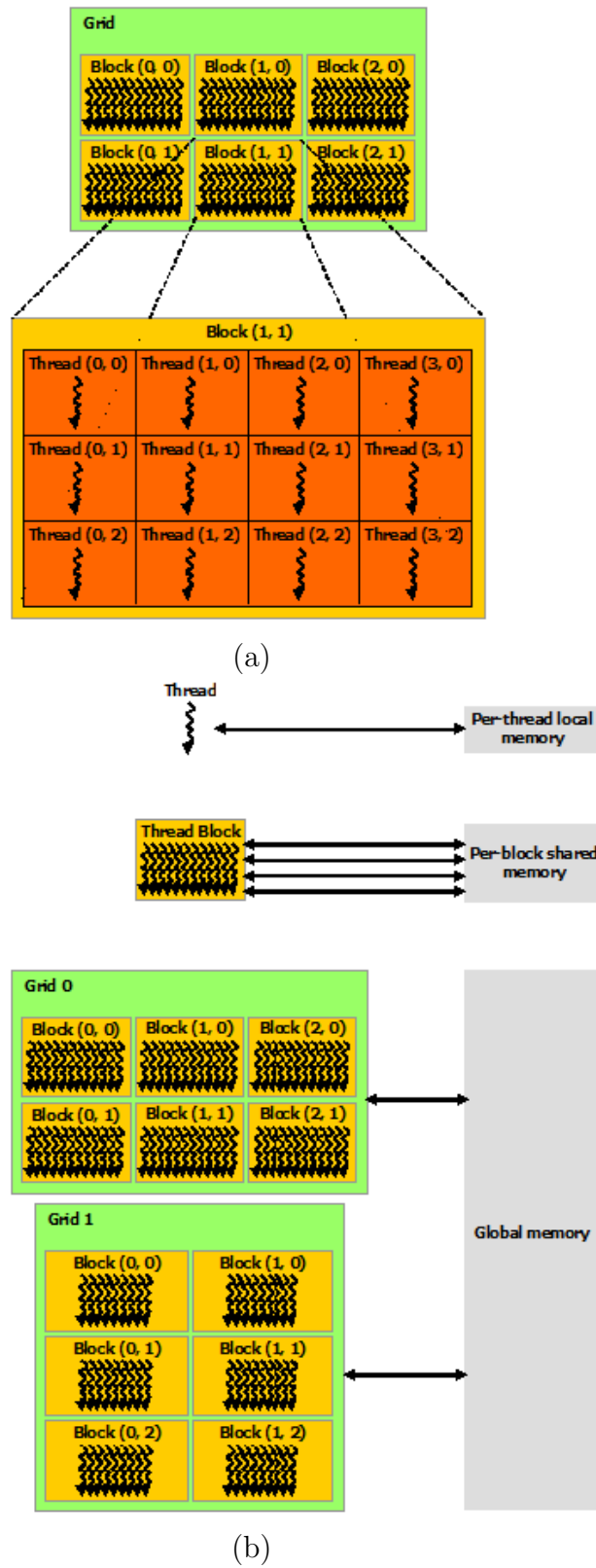


Figure 4.2: GPU threads blocks and memory: (a) threads and block structure; (b) CUDA memory hierarchy. Taken from reference [110].

threads. To allow threads to access a larger memory space quickly, two additional read-only memories are available, constant memory and texture memory, which use a cache for quick access. The memory hierarchy is shown in Figure 4.2(b). Since the GPU memory resides physically on the card, transfer of memory from the CPU to the GPU, and vice-versa, is restricted by the PCIe bus speeds, which are much slower than the on-board memory. The amount of memory transfer between the RAM (CPU memory) and the GPU is the main bottleneck in GPU computations and *should be minimised* as much as possible. One way of doing so is to assign memory transfer during large computational tasks so that the memory transfer is essentially hidden.

A good GPU code should make clever use of the different memory spaces, minimising memory transfer, and optimising the threads per block so as to utilise as much of the GPU capacity as possible. For more information on GPU architecture and programming with CUDA, the reader is referred to [110].

It is worth adding that each SM has dedicated single precision floating point 32 bits (FP32) cores and double precision floating point 64 bits (FP64) cores that require double the memory, with massive differences in the number of FP64 cores per SM in all but a few gaming cards. This leads to huge differences in the ratio of single to double precision efficiency between hardware, with regular GTX gaming cards having a huge 1:24 or 1:32 ratio between FP32 and FP64 (with few exceptions) while professional GPGPU cards, costing up to 5 times more than an equivalent gaming card, have a 1:3 (and very recently 1:2) ratio. Due to this restriction, the use of single precision wherever permitted is highly desirable for the sake of computational speed.

4.2 SPH on GPUs

The main iteration of a SPH algorithm contains three separate steps: neighbour searching; particle interaction; and system update (time-stepping). In order to implement an efficient code for GPUs each of these steps require careful consideration.

Crespo [30] presents an analysis of the most suitable algorithms and techniques to take advantage on the GPU capabilities. He recommends having the whole problem reside on the GPU, with the CPU only initialising the data and passing it to the GPU at the start of the simulation, and offloading data back to the CPU for output purposes every number of time steps, thus reducing memory transfer. Additionally,

he suggests using constant memory for all constant values (such as initial density, mass etc.) and shared memory for computations inside each block, thus limiting access to the global GPU memory, employing reduction techniques on the GPU, and aligning information passed to the GPU in arrays.

In mesh-less methods, such as SPH, where data points (particles) move with the flow, there is no fixed connectivity as there is in grid-based methods. Neighbours particles must be found for each particle whenever their positions are updated. This is a dynamic rather than static problem which requires a neighbour searching algorithm. Domínguez et al. [39] highlighted the importance of neighbour searching and provided an optimised searching algorithm for CPUs which uses a linked list of neighbouring cells and a reordering of particles in a way that particles in adjacent cells are close in memory. This provides a more regular memory access pattern which decreases memory access times and hence computational efficiency. This technique was later adapted to GPUs [37].

In the discrete sense, particle interaction calculations in SPH can be thought of as an n-body problem, where each particle interacts with another in a pair-wise manner and is independent from all others (in space). The pairwise interaction means that no matrices (at least in the basic SPH formalism) are required for the force calculations. This allows for a more efficient memory usage on the GPU. The parallelisation of the interactions is done “per interaction”, rather than “per particle”, i.e. each summation over neighbouring particles is carried out in parallel with the pair-wise interaction done in a serial sweep, rather than each summation done in serial with the pair-wise interaction done in parallel. This approach minimises the number of threads and allows for data re-usability.

Explicit time-stepping is very straight forward to perform in parallel. Each particle requires only the data calculated in the particle interaction step, performed independently on a per-particle basis, thus requiring no communication between particles - a very important feature for GPU parallelisation. Following [30] the only scope for optimising the time-stepping scheme for the GPU is the calculation of the time-step restrictions, which require knowledge of the maximum forces and velocities of the particles. This can be performed by an efficient reduction algorithm which utilises the shared memory per block.

Several SPH GPU codes are available in the open domain, such as GPUSPH [62] and DualSPHysics [28], both of which are derivatives of SPHysics [51] and implemented using CUDA, and AQUAgpusph [19] which is implemented using OpenCL.

At the start of this work, only DualSPHysics was mature enough to provide a solid working environment in which development of new features seemed feasible, and hence was the code adopted for modification. A detailed description of DualSPHysics and the modifications to the original source code introduced in this work are provided in the following sections.

4.3 DualSPHysics open-source code

DualSPHysics is a highly optimised CPU and GPU code written in C++ with OpenMP for CPU parallelisation and CUDA for GPU parallelisation. The code was developed by a collaboration between the EPhysLab in Universidade de Vigo (Spain) and MACE in the University of Manchester (UK), and is based on the FORTRAN SPHysics code, with which it shares the formulation and many its features. The version of DualSPHysics at the time of writing is v4.0. However, the previous version, v3.1, was used for much of the work conducted during the course of this project. One of many differences between v3.1 and v4.0 is the inclusion of support for double precision, which was omitted in all versions of DualSPHysics prior to v4.0. As will be shown later, precision can play an important role in the ability to obtain correct results. In addition to the open-source SPH solver, DualSPHysics ships with a pre-compiled preprocessor (called GenCase) and various post-processing tools (with PartVtk and MeasureTool the most important ones) as well as a very simple open-source post-processing tool called ToVtk.

DualSPHysics was developed with the aim of exploring the potential efficiency of SPH, and only the most efficient algorithms (while not necessarily the most accurate) were included. Details of the neighbour searching algorithms can be found in reference [39] and [37]. Particle interactions follow the *direct density* formulation (3.40–3.41). Two options for time-stepping are provided, which are the velocity-Verlet scheme [51], and the symplectic scheme (3.82), the only one used in this work. Up to and including version 4.0, DualSPHysics provided only one type of wall boundary condition: the dynamic boundary condition [29] which was briefly described in Section 3.4.1. Two types of numerical dissipation are available: the artificial viscosity formulation (3.61) and the Molteni formulation of δ -SPH (3.68–3.70). The full list of available features in DualSPysics, which contains moving walls, floating bodies and several others can be consulted in reference [28]. Lastly, it is noted that for the purposes of this work only the GPU implementation description is provided.

The DualSPHysics code (GPU version) solves all three steps of the SPH iteration entirely on the GPU, with only initialisation and the occasional data output requiring data transfer from CPU to GPU or vice-versa, as explained in Figure 4.3. A diagram of the function calls in a typical SPH run is depicted in Figure 4.4.

To save memory transfer, the only variable arrays which are passed from the CPU to the GPU and back are the position, density and velocity arrays, which are also the only variables available as output. All other variable arrays exist only in the GPU, and are initialised at the start of every particle call to the `Interaction_Forces` subroutine by `PreInteraction_Forces`. These include the acceleration, density derivative and any other variable used for other features such as numerical dissipation. Since these subroutines are called after the neighbour searching and particle reordering, no reordering is required for any of these arrays, because the reordering is applied only to the variables which are kept between iterations.

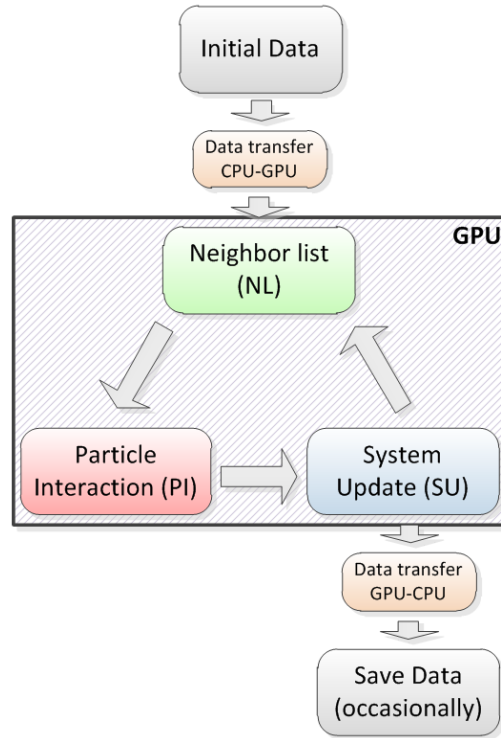


Figure 4.3: Flow diagram of the GPU implementation.

RUN	Starts simulation.
SelecDevice	Initialises CUDA device.
LoadConfig	Loads the configuration of the execution.
LoadCaseParticles	Loads particles of the case to be processed.
ConfigConstants	Configures value of constants.
ConfigDomain	Configuration of the current domain.
AllocGpuMemoryFixed	Allocates memory in GPU of for arrays with fixed size.
AllocGpuMemoryParticles	Allocates memory in GPU of main data of particles.
AllocCpuMemoryParticles	Allocates memory in CPU of main data of particles.
ReserveBasicArraysGpu	Arrays for basic particle data in GPU.
ParticlesDataUp	Uploads particle data to the GPU.
ConstantDataUp	Uploads constants to the GPU.
ConfigBlockSizes	Calculates optimum blocksize.
RunCellDivide	Generates neighbour list.
ConfigRunMode	Configures execution mode in GPU.
InitRun	Initialisation of arrays and variables for the execution.
PrintAllocMemory	Visualizes the reserved memory in CPU and GPU.
SaveData	Generates file with particle data of the initial instant.
MAIN LOOP	Main loop of the simulation.
ComputeStep_Sym	Computes particle interaction and system update using symplectic.
PREDICTOR	Predictor step.
Interaction_Forces	Call for particle interaction (PI).
PreInteraction_Forces	Prepares variables and assigns memory for PI.
Interaction_Forces	Computes particle interaction.
DtVariable	Computes the value of the new variable time step.
ComputeSymplecticPre	Computes System Update using
PosInteraction_Forces	Memory release of arrays in GPU.
RunCellDivide	Generates neighbour list.
CORRECTOR	Corrector step.
Interaction_Forces	Call for particle interaction.
PreInteraction_Forces	Prepares variables and assigns memory for PI.
Interaction_Forces	Computes particle interaction.
DtVariable	Computes the value of the new variable time step.
ComputeSymplecticCorr	Computes system update using symplectic=corrector.
PosInteraction_Forces	Memory release of arrays in GPU.
RunCellDivide	Generates Neighbour List.
SaveData	Generates file with particle data in specified interval.
FinishRun	Shows and stores final overview of execution.

Table 4.1: Simplified overview of the main functions called in DualSPHysics (version 4.0) GPU code using the symplectic time-stepping scheme.

4.4 DualSPHysicsGPM code

As explained above, DualSPHysics was developed with the most efficient formulations in mind, however it was soon realised that these formulations significantly lack in accuracy for the type of problems studied here. To address this deficiency several modifications were introduced to the latest version of the GPU implementation of the code, which at the time of implementation was the beta version of DualSPHysics v4.0. The modified version of the code is referred to in the remainder of this thesis as DualSPHysicsGPM, where GPM stands for ghost particles method. It is noted that the CPU implementation has not been modified in any way.

The modifications include the addition of the fixed ghost particles wall boundary condition described in Section 3.4.2 in combination with a no-slip BC using both laminar viscosity formulations described in Section 3.5.2, the corrected δ -SPH scheme described in Section 3.5.3, and both the continuity and inter-particle averaging formulations for the SPH particle interactions described in Section 3.1.6. All these changes are included in the particle interaction step of the SPH iteration, with the neighbour searching not needing any modification, and only a very minor adjustment required in the time stepping scheme. A detailed description of how each of these features were implemented and interact with the original code is given in the following sections.

In addition to the modifications to the SPH solver, pre- and post-processing tools were added and adapted. A very simple in house pre-processor was developed to generate and initialise particles in a more desirable way for the fixed ghost particle boundary condition (GPBC) method. This pre-processor is extremely limited and can only produce the few shapes required for this work. Despite that, it made the initialisation processes far simpler than using GenCase, the geometry pre-processor of DualSPHysics. For post-processing, the open source ToVtk code was adapted to include all features described in Section 3.8.

4.4.1 GPBC implementation

The main difference between the implementation of the GPBC and that of the dynamic boundary condition (DBC) is that the boundary particles are treated differently from the fluid particles. This required changing the order of several functions calls and the addition of several others. For ease of implementation, it was decided to split the code early in the function call sequence by introducing an additional

Interaction_Forces subroutine named Interaction_Forces_GPBC which is called with a conditional statement from the ComputeStep subroutine.

The first stage in the GPBC is to calculate the pressure of the wall particles via the force balance, equation (3.57), and the velocity interpolated from the fluid particles, equation (3.60). As these values are required prior to the fluid-boundary interactions, the GPU kernel KerInteractionForcesBound was moved between PreInteraction_Forces and Interaction_Forces_GPBC subroutine, and are stored in two additional arrays of type float and float3 (a structure containing a triplet of FT32 numbers) respectively, which are stored in the GPU only. After their calculation, an additional subroutine PreInteractionForcesBound is called to copy these values into the existing density array by the reverse equation of state (3.58) and velocity array. This way, the only extra work required arises from the increased number of boundary particles and from a minimal amount of additional instructions required for copying them into the existing arrays.

As the wall pressure, and hence density, are obtained through a summation over the neighbourhood fluid particles and not advanced in time, the time-stepping algorithm has been modified to exclude advancing the boundary particles in time.

4.4.2 Corrected δ -SPH implementation

The implementation of the corrected δ -SPH scheme proposed by Antuono et al. [8] is significantly more involved than that of the GPBC. It requires two additional summations (and hence separate loops) over the fluid particles, and the calculation of the inverse of the correction matrix, or preferably the solution of the $\mathbf{Ax} = \mathbf{b}$ matrix-vector system. Both these loops take place between the PreInteraction_Forces and the Interaction_Forces subroutines, as seen in Figure 4.5. In addition, the nature of the correction matrix, which is calculated by the moving least-squares method (MLS) is such that it can have a mixture of very large and very small numbers. A quick test showed that single precision is not adequate to accurately solve the matrix-vector system, and hence its solution is obtained using double precision. This not only requires a significant amount of extra memory on the GPU, but also significantly slows down the computation due to the GPU architecture performing far worse in double than in single precision.

The correction matrix in equation (3.73) is computed in the first loop, where a full 3×3 matrix (for both two and three-dimensional cases) in double precision is stored. Storing such a large matrix in one array is very inefficient in the GPU, however, it

is the approach chosen here for simplicity. After calculating the correction matrix, a *LU* decomposition with row pivoting is performed, with the results stored in BLAS notation [107] by overwriting the original matrix. As the BLAS notation does not conserve the pivoting information, a secondary unsigned array is passed with a single digit between 0 to 5, which describes the pivoting performed in the following way: 0 indicates no pivoting; 1 indicates the first and second row exchanged; 2 indicates the first and third row exchanged and so on. An additional digit, 6, indicates that the determinant of the matrix, which is calculated during the *LU* factorisation, is below a threshold of 10^{-8} .

In the second loop, the corrected density gradient, equation (3.72), is computed and stored in an array of `float3`. At the start of the loop, the pivoting array is checked to see if the determinant is valid and, if so, a backward-forwards substitution solver is called. The pivoting information is checked in the solver, and the solution vector is reordered accordingly. The calculations inside the solver are performed in double precision, and the results converted to single when saved in the corrected density gradient array. This array is then passed to the main `InteractionForcesGPBC` routine, where it is calculated in the same manner as the Molteni δ -SPH formulation, equation (3.69).

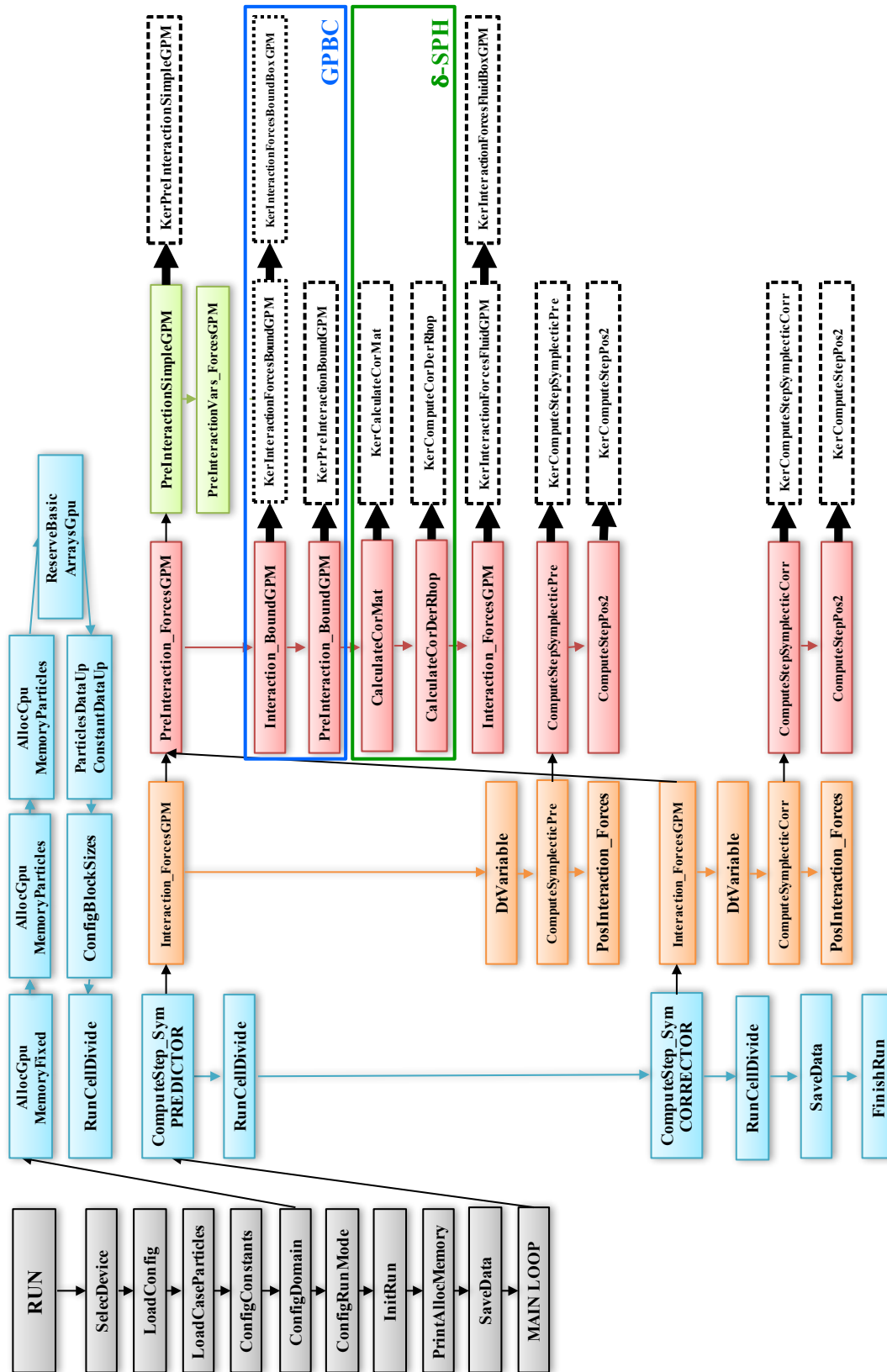


Figure 4.5: Call function graph for DualSPHysicsGPM.

RUN	Starts simulation.
SelecDevice	Initialises CUDA device.
LoadConfig	Loads the configuration of the execution.
LoadCaseParticles	Loads particles of the case to be processed.
ConfigConstants	Configures value of constants.
ConfigDomain	Configuration of the current domain.
AllocGpuMemoryFixed	Allocates memory in GPU of for arrays with fixed size.
AllocGpuMemoryParticles	Allocates memory in GPU of main data of particles.
AllocCpuMemoryParticles	Allocates memory in CPU of main data of particles.
ReserveBasicArraysGpu	Arrays for basic particle data in GPU.
ParticlesDataUp	Uploads particle data to the GPU.
ConstantDataUp	Uploads constants to the GPU.
ConfigBlockSize	Calculates optimum blocksize.
RunCellDivide	Generates neighbour list.
ConfigRunMode	Configures execution mode in GPU.
InitRun	Initialisation of arrays and variables for the execution.
PrintAllocMemory	Visualizes the reserved memory in CPU and GPU.
SaveData	Generates file with particle data of the initial instant.
MAIN LOOP	Main loop of the simulation.
ComputeStep.Sym	Computes particle interaction and system supdate using Symplectic.
PREDICTOR	Predictor step.
Interaction_ForcesGPM	Call for particle interaction (PI).
PreInteraction_ForcesGPM	Prepares variables and assigns memory for PI.
Interaction_BoundGPM	Computes boundary-fluid PI.
PreInteraction_BoundGPM	Copies boundary particle data to main arrays.
CalculateCorMat	Calculates the correction matrix.
ComputeCorDerRhop	Computes the corrected density gradient.
Interaction_ForcesGPM	Computes particle interaction.
DtVariable	Computes the value of the new variable time step.
ComputeSymplecticPre	Computes system update using predictor step.
PosInteraction_Forces	Memory release of arrays in GPU.
RunCellDivide	Generates neighbour list.
CORRECTOR	Corrector step.
Interaction_Forces	Call for particle interaction.
PreInteraction_ForcesGPM	Prepares variables and assigns memory for PI.
Interaction_BoundGPM	Computes boundary-fluid PI.
PreInteraction_BoundGPM	Copies boundary particle data to main arrays.
CalculateCorMat	Calculates the correction matrix.
ComputeCorDerRhop	Computes the corrected density gradient.
Interaction_ForcesGPM	Computes particle interaction.
DtVariable	Computes the value of the new variable time step.
ComputeSymplecticCorr	Computes system update using symplectic-corrector.
PosInteraction_Forces	Memory release of arrays in GPU.
RunCellDivide	Generates neighbour list.
SaveData	Generates file with particle data in specified interval.
FinishRun	Shows and stores final overview of execution.

Table 4.2: Simplified overview of the main functions called in DualSPHysicsGPM code using the symplectic time-stepping scheme and corrected δ -SPH.

4.5 Single versus double precision

As briefly discussed earlier in Section 4.1, the use of single precision (FP32) is highly desirable for GPU codes. Up to the latest version of DualSPHysics, only single precision was supported with no option for the use of double precision numbers (FP64). However, for some situations the use of single precision is not sufficient and various errors start appearing due to rounding error. Cases which are most affected are those where the fluid height to width ratio is large, where there is a very large domain, or simply when the computational domain is positioned far from the origin. The issue arises because the number of decimal places available for a particle's position in single precision on one side of the domain is very different to those available for a particle on the other side, which leads to very poor calculations.

To illustrate the problem, consider a simplified case with a with particles spacing of $\Delta x = 0.001\text{m}$. In this case a particle placed to the right of the origin has the x_a value of

```
1.0000000e-3 (single)
1.0000000000000000e-03 (double)
```

in single and double precision respectively. If the origin is shifted by 10 000, the x_a value of the particle becomes

```
1.0000001e+04 (single)
1.0000001000000000e+04 (double).
```

Now, if after a SPH iteration the particle is shifted by a fraction of Δx so that $x_{new} = x_a + 0.3\Delta x$, the value of the particle becomes

```
1.0000001e+04 (single)
1.0000001300000000e+04 (double)
```

where for single precision, due to the available decimal places, the shift in the particle position is simply not registered. In realistic cases the ratio of the particle position to the particle spacing is rarely this extreme, however the shift in position is usually not a simple value and can contain many decimals. This is enough to cause a significant portion of the shifting value to be missed and in turn drastically affect the calculation.

As a real example, the following case uses the geometry and discretisation from a simulation of an experiment described in Section 6.1. In this case $\Delta x = 0.0005\text{m}$ and the width of the domain is $L = 0.59\text{m}$ so that the ratio of maximal value of the position to the particle spacing is $x_{max}/\Delta x = 1180$ (which is a far smaller ratio

than the ratio of the simplified case above where $x_a/\Delta x = 10^7$). To lower the ratio further, it is possible to shift the origin to the centre of the fluid domain so that $x_{max}/\Delta x = 590$. To visualise the precision related errors, both these cases (with the origin positioned at the lower left corner and the origin positioned at the centre of the tank) were run for a duration of one second using single precision. Figure 4.6 shows that both cases have a spurious pressure field, with the errors growing stronger further away from the origin. In the case with the origin positioned in the bottom left corner (top figure), the left side of the tank is relatively error free while the right side of the tank has very noisy values. In contrast, for the case where the origin is centred (bottom figure), the centre of the tank shows a relatively small error while further away the errors are symmetric around the origin and are significantly smaller than in the previous case.

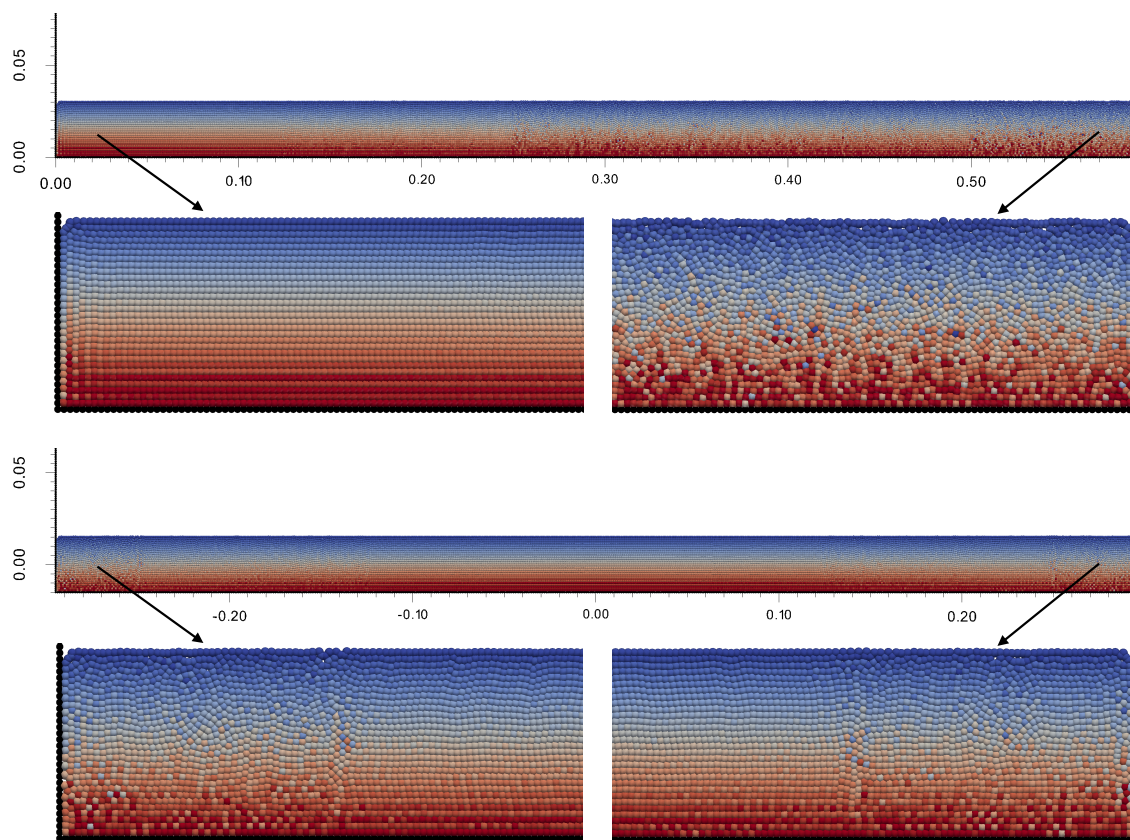


Figure 4.6: Single precision errors in the pressure field for different positions of the origin in a stretched domain after simulating one second: (top) origin placed at bottom left corner; (bottom) origin placed at centre of fluid domain.

In order to counter this issue, a study of different approaches to implement in

DualSPHysics are explored in reference [37]. The first approach is to change the code to use double instead of single precision everywhere. The second is to use double precision only for the particle positions and the operators for updating the variable, while keeping everything else (such as density, velocity and their derivatives) in single precision. The last option is to use a relative coordinate system, where all operations are done in single precision from a position centred on the neighbour searching cells. The positions are then mapped back to the global coordinates for neighbour searching, using the know cell position in the domain.

The author concluded that the first approach is very expensive, while the other two exhibited only minor differences. Using double precision only for the positions and updating of variables is slightly less efficient than using a relative coordinate system. However it allows for a much simpler code design and maintainability. These features are critical to an open-source code and hence the author of reference [37] decided to implement double precision for the positions only in the latest release of DualSPHysics (v4.0). From this point onwards, this is the double precision approach adopted in this thesis.

4.6 Performance of DualSPHysicsGPM

To assess the performance of the DualSPHysicsGPM code, a simple test case is conducted comparing DualSPHysics using the dynamic boundary condition (DBC) against DualSPHysicsGPM using the fixed ghost particles boundary condition (GPBC) and against the GPBC with the addition of the corrected δ -SPH scheme. The test consists of a cubic enclosed tank with a side length of 1m and filled with fluid. Particle spacings $\Delta x = 0.001\text{m}$ in 2D and $\Delta x = 0.01\text{m}$ in 3D were chosen, which give rise to 998 001 and 970 299 fluid particles, respectively. The DBC simulation has 4 000 boundary particles in 2D and 60 002 in 3D, while the GPBC case has 12 024 and 197 326 respectively.

The computer used for this test has an i7 5960X octa-core CPU (16 threads) over-clocked to 3.9 GHz, 32GB of DDR4 2666 MHz RAM and a Geforce GTX Titan Z GPU. The GTX Titan Z is the very top end gaming card from the previous generation GPUs, code-named Kepler, which includes 2 GPUs on one card. Each GPU has 15 SM with 2880 CUDA cores, 6GB of DDR5 memory and a base clock speed of 0.88 GHz. However, when only one of the two GPUs is running, the clock rate is boosted to 1.1 GHz, as was the case in these tests. The GPU chips on the

GTX Titan Z (named GK210) are identical to the chip on the top of the line Tesla K80 GPGPU, with only a few features omitted from the GTX line. This GPU has a single to double precision ratio of 1:3, making it far more suitable for double precision calculations than most other gaming cards.

Each case was run in both single (FP32) and double (FP64) precision and their performance evaluated in terms of the number of steps per second (SPS). These results are summarised in Table 4.3.

Table 4.3: Comparison of performance for different features of the code. Steps per second (SPS) are compared between the dynamic boundary conditions (DBC), the fixed ghost particle boundary condition (GPBC) and the GPBC with the corrected δ -SPH scheme in both two and three dimensions and using either single or double precision.

Test type	DBC	GPBC	GPBC + δ -SPH
2D Single	SPS: 53.1	SPS: 50.9	SPS: 10.2
2D Double	SPS: 46.5	SPS: 40.2	SPS: 8.9
3D Single	SPS: 11.1	SPS: 12.3	SPS: 1.8
3D Double	SPS: 8.4	SPS: 8.9	SPS: 1.5

The results show that the GPBC performs as well as the DBC with only a minor difference in the SPS between the cases. The number of SPS in the GPBC with δ -SPH is reduced by a factor between approximately 4.5 to 7 with respect to either the DBC or the GPBC, depending on the specific case. This significant increase in computational cost is due to the two additional loops over the fluid particles that use double precision required to compute the correction matrix and solve the linear system in the δ -SPH scheme.

This first test uses a very large number of particles in order to fill up all the resources in the GPU. However, SPH simulations are often performed on far smaller scales with only thousands of particles. In such cases, as a large portion of the GPU's resources are free, some of the FP64 operations can be performed on the free cores, which lead to a significant reduction in the penalty. To verify this, the test is run again in 2D using $\Delta x = 0.001\text{m}$. This set-up gives rise to 9 801 fluid particles with 400 and 1 224 boundary particles for the DBC and GPBC formulations respectively. The results are given in Table 4.4.

In this case the δ -SPH scheme is far more affordable, with the SPS reduced to just under a factor of 2 compared to either of the two other cases. This indicates that despite the huge costs of the scheme, the maximal overheads will not be reached in

Table 4.4: Comparison of performance for different features of the code for a small scale two dimensional case. Steps per second (SPS) are compared between the dynamic boundary conditions (DBC), the fixed ghost particle boundary condition (GPBC) and the GPBC with the corrected δ -SPH scheme using either single or double precision in a Nvidia GTX Titan Z GPU card.

Test type	DBC	GPBC	GPBC + δ -SPH
2D Single	SPS: 735	SPS: 715	SPS: 430
2D Double	SPS: 690	SPS: 670	SPS: 380

many common cases, making the scheme far more attractive. It is however important to note that this reduction in the cost depends greatly on the hardware used and such savings will not be possible with older GPUs or those that do not provide good FP64 support. To illustrate this latest point, the same case was run again using a Nvidia GTX 970 with the same workstation. The GTX 970 is a high-end gaming GPU from the latest generation GPU architecture, code named Maxwell, which has 13 SM with 1668 CUDA cores, 4GB of DDR5 memory and clock speed of 1.33GHz. This card is optimised for gaming and, like all Maxwell processors, has a very bad single to double precision ratio of 1:32. Table 4.5 shows that the use of the δ -SPH scheme increases the computational cost by around 3.5 times with the GTX 970 – almost double to the computational cost difference using the GTX Titan Z – despite not using most of the SM.

Table 4.5: Comparison of performance for different features of the code for a small scale two dimensional case. Steps per second (SPS) are compared between the dynamic boundary conditions (DBC), the fixed ghost particle boundary condition (GPBC) and the GPBC with the corrected δ -SPH scheme using either single or double precision in a Nvidia GTX 970 GPU card.

Test type	DBC	GPBC	GPBC + δ -SPH
2D Single	SPS: 960	SPS: 940	SPS: 285
2D Double	SPS: 850	SPS: 815	SPS: 275

To conclude, it has been shown that the use of the GPBC has negligible effect on the run times compared to the original code, while the corrected δ -SPH incurs a significant increase in computation cost: up to a seven-fold increase. The additional cost of the δ -SPH scheme however is strongly affected by the dimension of the problem, the number of particles and the hardware used, dropping in some scenarios to only a two-fold increase. Optimisation of the scheme is hence strongly desirable but is left for future work.

Chapter 5

Verification and validation

This chapter presents a variety of tests conducted to verify and validate the modified DualSPHysics code, which will be referred to in the following as DualSPHysicsGPM. This code includes the fixed ghost particle boundary condition of Adami et al. [3] described in Section 3.4.2 and the corrected δ -SPH scheme of Antuono et al. [8] described in Section 3.5.3. As well as verifying (i.e. checking the equations are solved correctly) and validating (i.e. that the equations being solved model the physical problem correctly), it will also show how the modifications in DualSPHysicsGPM improve on DualSPHysics in several ways. These include reducing energy dissipation, providing a smoother density field, and avoiding non-physical gaps between the fluid and boundary associated with the use of the dynamic boundary condition (DBC).

The verification section includes two tests which have analytical solutions. The first is a hydrostatic test case (still water) which is run for a long period of time. The hydrostatic test case is commonly used for verification in the SPH literature as it allows to carefully examine numerical issues which are harder to quantify in dynamic cases, as explained in Section 3.2. The second verification test is a linear sloshing case which has analytical solutions obtained from potential flow theory [43] and is the most relevant verification test case for the topics investigated in the thesis.

The validation is carried out by comparing SPH simulations against experiments and published results in two and three dimensions. The two-dimensional test case is the classical dam break simulation. This case is based on an experiment by Buncher [16] and is used in several publications, including those used to validate the additional features in DualSPHysicsGPM [3, 8]. This allows for comparison against published results using the same features as well as the experimental data. The three-dimensional test case is a dam break in a tank with a box. This test is a

common validation test for SPH in three dimensions and is a benchmark suggested by the SPHERIC community [70]. It is often used by DualSPHysics developers for validation of new features [38]. The fluid flow exhibits violent impact with the walls, splashing and overturning waves which are all features SPH is well suited to model.

5.1 Hydrostatic simulations

The hydrostatic tank simulation is one of the simplest cases for CFD code verification. However, this test is extremely challenging for SPH due to various numerical errors, as explained in Section 3.5. These difficulties mean that obtaining a correct hydrostatic pressure profile with weakly compressible SPH is a difficult task. This test case allows for detailed comparison between the different boundary conditions and dissipation schemes by studying differences in the pressure field against the hydrostatic pressure and monitoring energy conservation.

The implementation of the ghost particle boundary condition (GPBC) in DualSPHysicsGPM, as described in Section 3.4.2, is verified first and compared against the published results from Adami et al. [3]. It is then compared against the dynamic boundary condition (DBC) available in the DualSPHysics code. The test is also used to evaluate the ability to preserve volume and conserve energy of the different dissipation schemes.

5.1.1 Hydrostatic boundary condition verification

The fixed ghost particle boundary condition, as described in Section 3.4.2 and implemented in the DualSPHysicsGPM code, is compared against results by Adami et al. [3] for the still water case described in the paper.

The problem consists of a two-dimensional open roof rectangular tank. The quantities that follow are non-dimensionalised by a characteristic length scale $l = 1\text{m}$, a density of water $\rho = 1000\text{kg/m}^3$, and reference velocity $v = 2.97\text{m/s}$. The length of the tank is $L = 2$, with fluid filled up to a height $H = 0.9$ with the values of density, velocity and acceleration due to gravity given as $\rho_0 = 1$, $v_{\max} = \sqrt{|gH|} = 1$ and $\mathbf{g} = (0, g)$ with $g = -\frac{10}{9}$, respectively. Following the paper, particle spacing is set to $\Delta x = 0.02$ which gives 4500 fluid particles. The artificial viscosity coefficient (3.61) is set to $\alpha = 0.24$ and the numerical speed of sound is $c_0 = 10v_{\max}$. The simulation was run up to a time $t = 2t_{\text{ref}}$, where $t_{\text{ref}} = H/v_{\max}$ with a warm-up stage, defined in equation (3.74), over a damping period of $t = t_{\text{ref}}$. The pressure profiles

are compared against the analytic values of the hydrostatic pressure. Figure 5.1 shows that the results from DualSPHysicsGPM agree very well with the hydrostatic pressure profiles and the published results.

It is important to note that the amount of artificial viscosity here is very high. Using the relation (3.63), as done in reference [3], $\alpha = 0.24$ corresponds to an “effective” Reynolds number of $Re = 100$. Using such a high viscosity value strongly damps most numerical errors, but it would not be suitable for dynamic cases because it will seriously affect the dynamics of the flow. The only conclusion to draw from this test is that the implementation in DualSPHysicsGPM agrees with the published results. Hence, an additional investigation is required in order to further verify the accuracy of the boundary condition. This is done in the following tests.

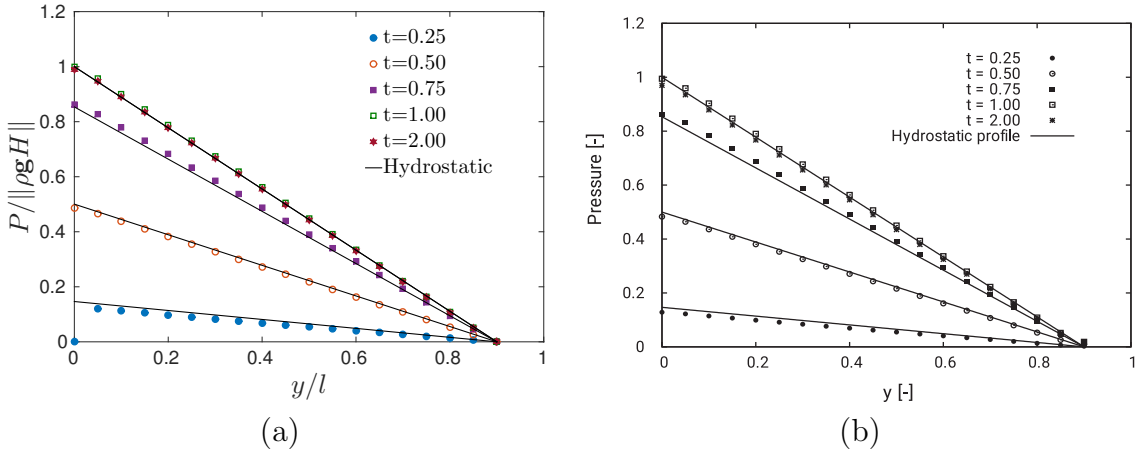


Figure 5.1: Pressure profiles at the centre of the fluid domain normalised by $\|\rho g H\|$ against the fluid height y normalised by l during the warm-up stage using a fixed ghost particle boundary condition: (a) using the ghost particle boundary condition implementation in DualSPHysicsGPM; (b) plot taken from [3].

5.1.2 Hydrostatic boundary condition comparison

A long duration, low artificial viscosity, test is examined in detail to compare the differences between the dynamic boundary condition (DBC) in DualSPHysics and the fixed ghost particle boundary condition (GPBC) from Section 3.4.2 in DualSPHysicsGPM. This test case consists of a square two-dimensional tank of length $L = 1\text{m}$ filled with water to a height $H = 0.5\text{m}$. The particle spacing is set to $\Delta x = 0.01\text{m}$ which gives 5000 fluid particles. Fluid parameters are those of water, i.e. $\rho_0 = 1000\text{kg/m}^3$ and $\gamma = 7$, with the numerical speed of sound chosen as

$c_0 = 10v_{\max}$ where $v_{\max} = \sqrt{|gH|}$ and $\mathbf{g} = (0, g)$ with $g = -9.81\text{m/s}^2$ is the acceleration due to Earth's gravity. Only artificial viscosity (3.61) is used with $\alpha = 0.02$, which is a common value used for inviscid SPH simulations (see for example reference [8]). Both cases use the direct density formulation, (3.40–3.41), since it is the only available option in DualSPHysics. Each case is simulated for a period of 100 seconds starting from a hydrostatic initialisation. The seemingly long duration is essential to allow for the examination of the effects of boundary conditions after the initial reconfiguration of the fluid particles into a zero-energy state (see Section 3.5.4).

Figure 5.2 shows the pressure field at $t = 100\text{s}$ for both cases. Closer inspection shows the difference in the pressure values close to the boundaries, especially close to the bottom of the tank, as well as the distortion of the free surface with the DBC which can be seen to be slightly pushed away from the wall. Figure 5.3 shows the pressure value of all the particles normalised by the maximal hydrostatic pressure $P/\|\rho\mathbf{g}H\|$ against their normalised height y/H compared to the hydrostatic pressure profile. While both boundary conditions generally agree with the hydrostatic profile, especially towards the free surface ($y/H = 1$) the DBC shows a much wider spread, meaning that more particles deviate from the correct analytic hydrostatic pressure value.

To obtain a better understanding of the reasons for such behaviour, the evolution of the total energy and its components are examined. Figure 5.4 shows the evolution of total (a), kinetic (b), potential (c), and internal (d) energy over time, normalised by a reference energy $E_{\text{ref}} = \|\rho_0 H^3 \mathbf{g}\|$ [96].

In the DBC case, energy is far less accurately conserved, showing an order of magnitude drop in the total energy in comparison to the GPBC, as observed in Figure 5.4(a). The various energy components are analysed in order to identify the source of this difference and better understand what causes it. The kinetic energy, $E_{\text{kin}}(\mathbf{v})$, the external potential energy due to gravity $E_{\text{ext}}^{\mathbf{g}}(\mathbf{x})$ and the internal energy $E_{\text{int}}(\rho)$ are functions of the velocity, position and density respectively. This means that by analysing each of the energy components separately, the effect of the boundary condition on each of their dependent variables can be assessed.

The main cause for the drop in total energy comes from the potential energy, as seen in Figure 5.4(c). This is likely caused by the disordered distribution of particles at the free surface, which noticeably drop from their initial positions, as seen in Figure 5.2. The initial oscillations are also far larger using the DBC than the

GPBC, indicating an overall larger reshuffling of the particles positions, although both reach a steady value after around 40 seconds.

Another significant difference is observed in the kinetic energy, as seen in Figure 5.4(b), where the values from the DBC are an order of magnitude higher than those obtained by the GPBC. This indicates that the initial velocity is noticeably higher. In both cases, the kinetic energy gradually drops due to numerical diffusion, and reaches a value close to zero as the particles settle down into a lower energy state.

Finally, changes in the internal energy, which depends only on density, can be interpreted as a measure of compressibility. The internal energy oscillations, as seen in Figure 5.4(d), are always opposite to the potential energy oscillations and generally cancel each other out. This can be inferred from the far smoother time history of the total energy. Both boundary conditions reach a similar steady value, but a few noticeable differences are observed. The first observation is that the initial magnitude of the internal energy is far greater using the DBC, indicating that the initial pseudo-pressure waves are far stronger. The second observation is that despite the higher magnitude of the oscillations, the DBC dampens them faster. This damping might indicate that the boundary particles in the DBC absorb too much internal energy, and hence the values of density and pressure of the adjacent fluid particles are underestimated, also seen in Figure 5.2. Lastly, the internal energy value settled on after the initial oscillations obtained with the GPBC is slightly higher than that of the DBC. This indicated that the average density underwent slightly a larger increase using the GPBC than the DBC, making it seem more compressible. However, as explained above, the density values of the fluid particles adjacent to the solid walls are noticeably lower than the hydrostatic solution, meaning this conclusion might not be applicable to the bulk of the fluid domain.

To conclude, both the local properties of particles in the vicinity of the boundaries and the global energy conservation are evaluated more accurately with the GPBC than with the DBC. The GPBC, while not perfect, avoids many of the issues encountered by the DBC and it has a comparable computational cost, as shown in Section 4.6. The GPBC will be used for all further studies unless specifically indicated otherwise.

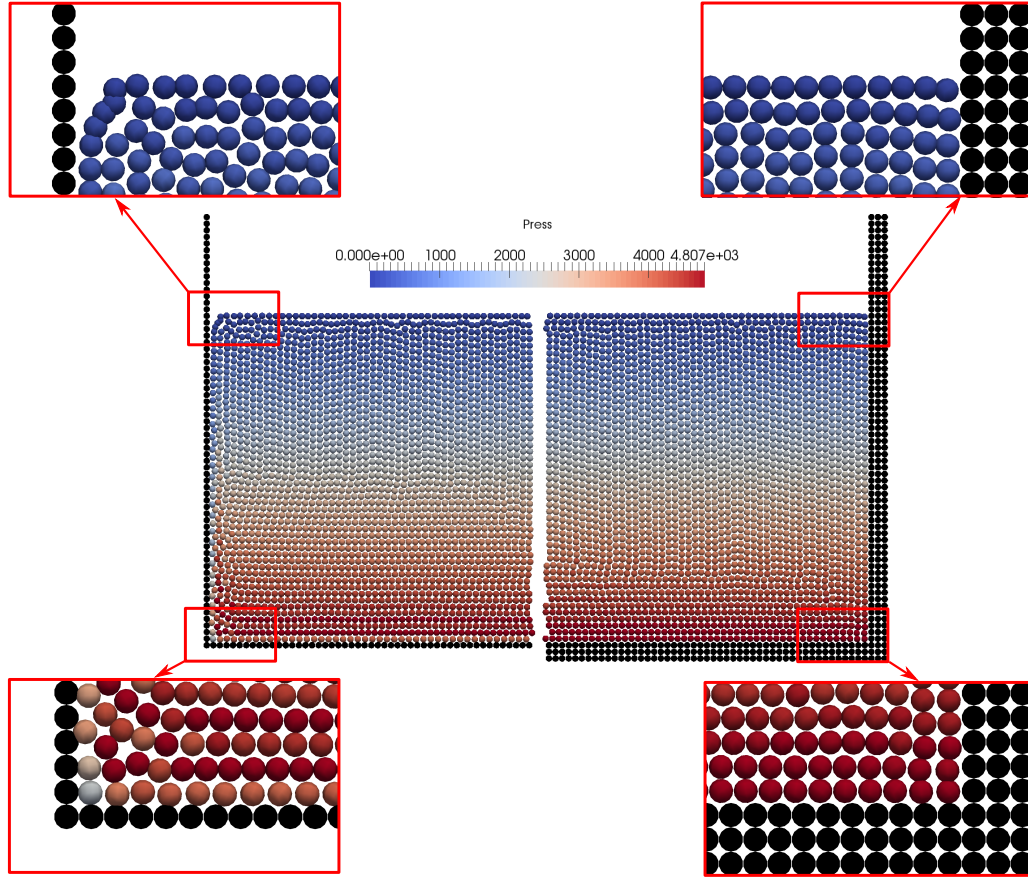


Figure 5.2: Pressure field at a hundred seconds of hydrostatic simulations using difference boundary conditions. (left) the dynamic boundary condition (DBC), (right) the fixed ghost particle boundary condition (GPBC).

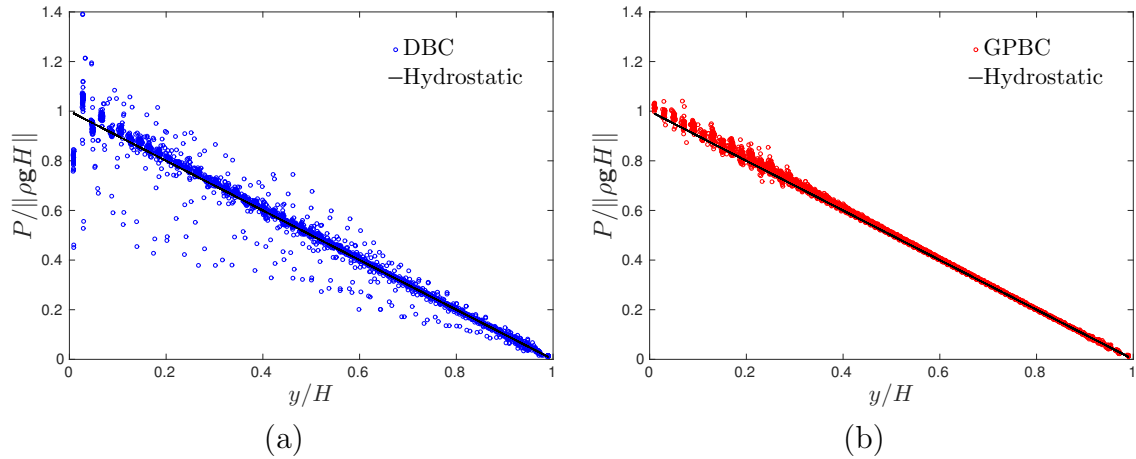
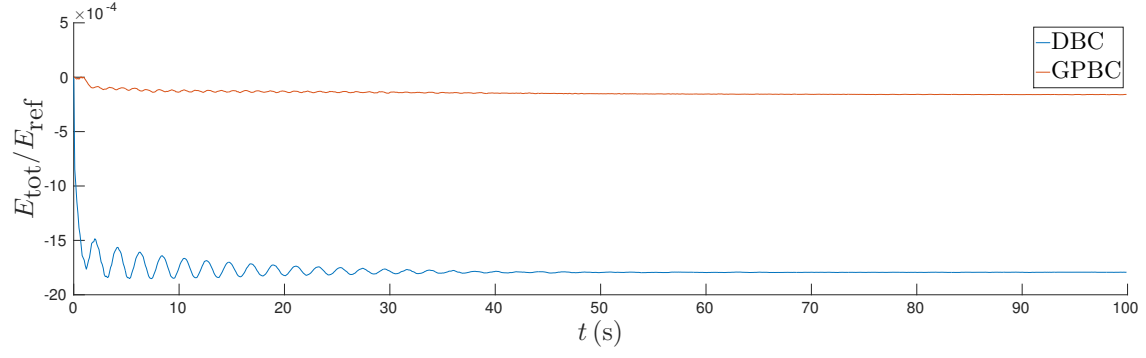
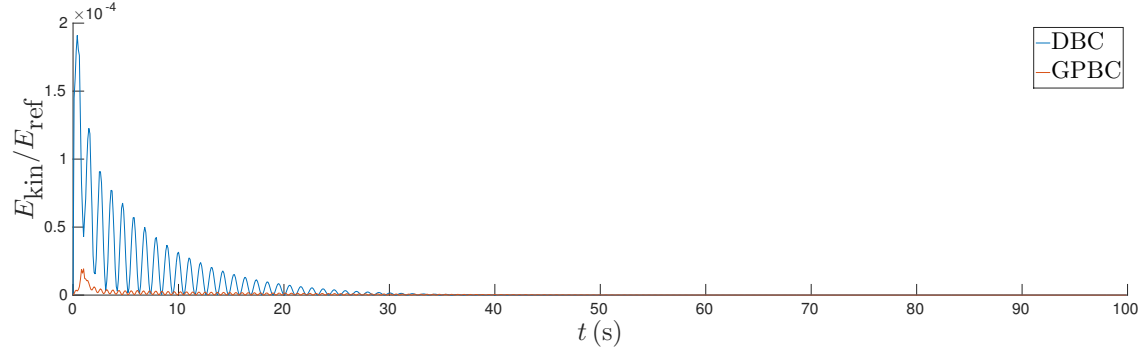


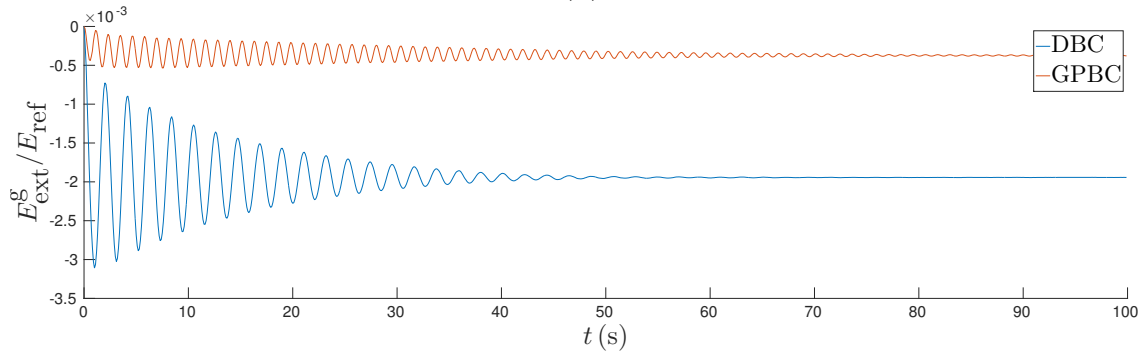
Figure 5.3: Pressure of all fluid particles normalised by $\|\rho\mathbf{g}H\|$ plotted against the particle's height (y) at a hundred seconds of hydrostatic simulations using different boundary conditions. (a) the dynamic boundary condition (DBC); (b) the fixed ghost particle boundary condition (GPBC).



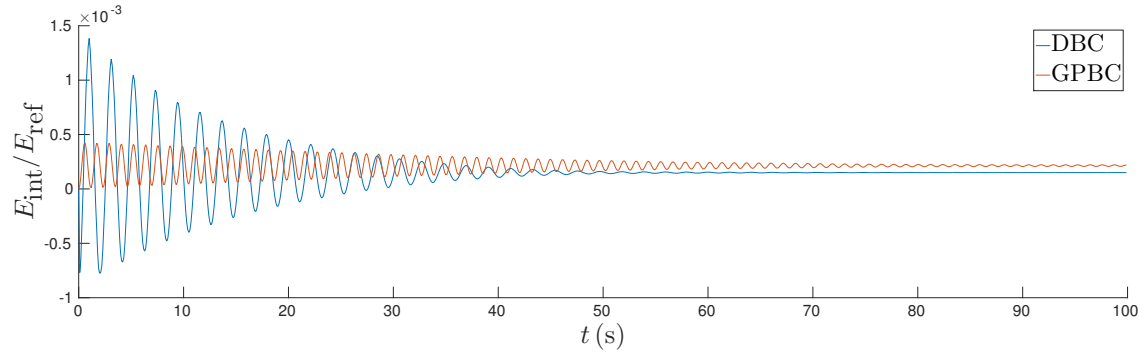
(a)



(b)



(c)



(d)

Figure 5.4: Energy components normalised by $E_{\text{ref}} = \|\rho_0 H^3 \mathbf{g}\|$ against time (s) of hydrostatic simulations using the dynamic boundary condition (DBC) and the fixed ghost particle boundary condition (GPBC). (a) total energy; (b) kinetic energy; (c) potential energy; (d) internal energy.

5.1.3 Hydrostatic δ -SPH comparison

The hydrostatic tank test case is used here to compare the different numerical dissipation schemes discussed in Section 3.5. These schemes are the Molteni et al. δ -SPH scheme [96], the Antuono et al. corrected δ -SPH scheme [8] and the artificial viscosity scheme [98]. These schemes will be referred to in the following as the Molteni, Antuono and AV schemes, respectively.

The same geometry and basic fluid parameters as in the previous still water case are used, being $L = 1\text{m}$, $H = 0.5\text{m}$, $\Delta x = 0.01\text{m}$, $\rho_0 = 1000\text{kg/m}^3$, $\gamma = 7$ and $c_0 = 10v_{\max}$ with $v_{\max} = \sqrt{|gH|}$ and $\mathbf{g} = (0, g)$ with $g = -9.81\text{m/s}^2$. Here the continuity formulation (3.38–3.39) is used for all cases, with both the Molteni and Antuono using $\delta = 0.1$ and no artificial viscosity, and the AV using $\alpha = 0.001$. The cases run for a duration of 100 seconds from a hydrostatic initialisation, allowing the simulation to develop over a prolonged duration.

The value of the artificial viscosity coefficient here, $\alpha = 0.001$, is much lower than the common value used, $\alpha = 0.02$. The reason for this is that $\alpha = 0.02$ proved to be too dissipative to compare against the δ -SPH cases with no artificial viscosity. Lower values of artificial viscosity were tested but were unstable, while this value led to a similar maximal value of kinetic energy. This indicates a similar level of dissipation of the velocity and hence provides a more balanced comparison.

Figure 5.5 shows the pressure field for all three cases at the end of the simulation ($t = 100\text{s}$). The Molteni scheme expands the particle spacing in the top part of the fluid domain, raising the fluid significantly above the initial height and higher than the other two cases. In contrast, the Antuono scheme keeps the fluid volume almost constant, with only a slight gap between the fluid and topmost layer of particles. The AV scheme similarly preserves the fluid volume and produces a more regular layer of particles at the top than the Antuono scheme, while the rest of the particles are slightly less ordered. This can be better realised in Figure 5.6 where the normalised pressure values are plotted against the heights along side the hydrostatic pressure profile for the AV and Antuono schemes. The AV scheme exhibits a *noisier* pressure field than the Antuono scheme. The total energy of the Molteni scheme, Figure 5.7, shows signs of continual growth rather than a constant value.

Further details are revealed by examining the first 10 seconds of the energy components, before the Molteni case starts to drastically expand in Figure 5.8. The kinetic energy has a similar magnitude for all schemes. Surprisingly, the AV simulation exhibits some oscillations. The potential energy calculated by the Molteni

scheme increases, as expected, as the fluid volume starts to expand. Both the Antuono and AV simulations tend to initially oscillate around a similar value, but the Antuono scheme effectively dissipates the oscillations within the first few seconds. The internal energy shows a similar pattern as the potential energy for the AV and Antuono schemes. Interestingly though, the Molteni scheme shows even more effective dissipation of the oscillations than the Antuono scheme, which might explain its popularity in dynamic short duration simulations.

The Molteni scheme generates non-physical fluid behaviour and therefore it must be ruled out as a candidate for long duration simulations. Hence the remainder of this section will focus on assessing the energy components of the Antuono and AV schemes exclusively. Figure 5.9 shows the evolution of the normalised total, kinetic, potential and internal components of the energy over time. Both schemes offer a similar ability to conserve the total energy, with the most noticeable difference being the nature of the oscillations. The Antuono simulation shows lower magnitude of oscillations but with a higher frequency than the AV scheme, but the AV scheme exhibits a higher magnitude with a lower frequency. This discrepancy is better analysed by looking at the other energy components. Looking at any of the individual energy components, the AV scheme is unable to damp these fluctuations, which persist throughout the whole duration of the simulation, while the Antuono scheme provides very effective damping and reaching a steady state very early. The potential energy of the Antuono scheme shows some noticeable high-frequency low-magnitude oscillations. It was noticed by inspection that the cause of these small oscillations is the reshuffling of the fluid particles at the top layer of the fluid, while the rest of the fluid domain remains unaffected. The reason the oscillations are restricted to the top layer of the fluid particles is yet to be determined and requires further investigation.

To conclude, the Molteni scheme causes critical failures and can not be considered as a viable method for numerical dissipation of long duration simulations. Both the AV and Antuono δ -SPH schemes show a similar ability to conserve the total energy. However, the AV scheme is unable to effectively damp the initial pseudo-pressure waves and these small oscillations remain in the computational domain for a very long time. On the other hand, the Antuono scheme induces high-frequency low-magnitude oscillations and a redistribution of particles on the free surface. This suggests that a higher viscosity value is required for the AV scheme to produce a smoother pressure field and damp the oscillations, which in turn would make the fluid more viscous than intended and affect the behaviour of the flow. Hence, the

Antuono scheme proves to be a better choice for numerical dissipation and is used exclusively to this effect for the remainder of the thesis unless otherwise stated.

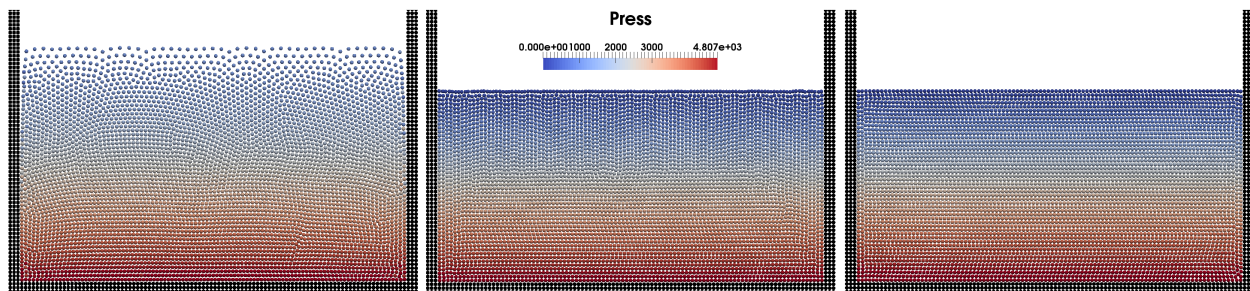


Figure 5.5: Pressure field at a hundred seconds of hydrostatic simulations using different numerical dissipation schemes: (left) Molteni δ -SPH, (centre) corrected Antuono δ -SPH, (right) artificial viscosity.

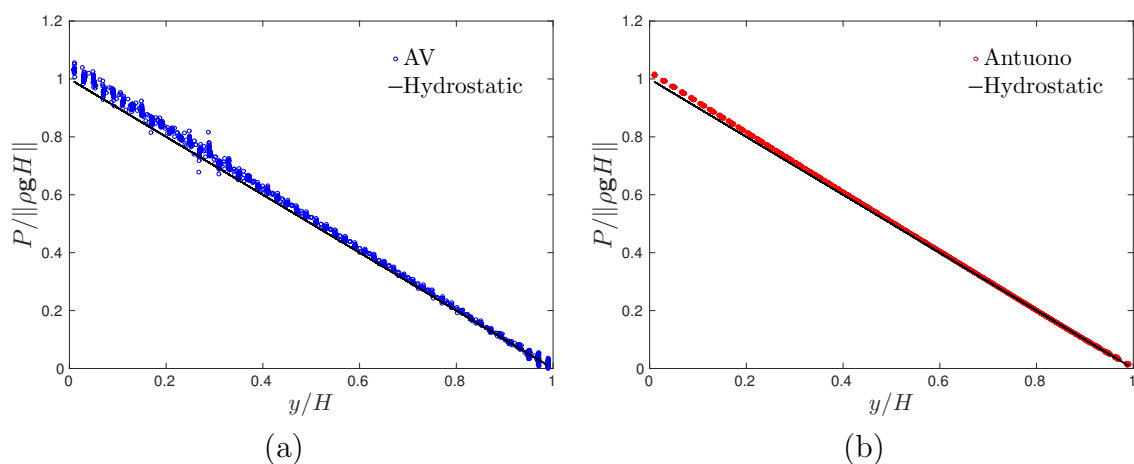


Figure 5.6: Pressure of all fluid particles normalised by $\|\rho \mathbf{g} H\|$ plotted against the particle's height (y) at a hundred seconds of hydrostatic simulations using different numerical dissipation schemes: (a) artificial viscosity (AV) with $\alpha = 0.001$; (b) corrected Antuono δ -SPH (Antuono) with $\delta = 0.1$.

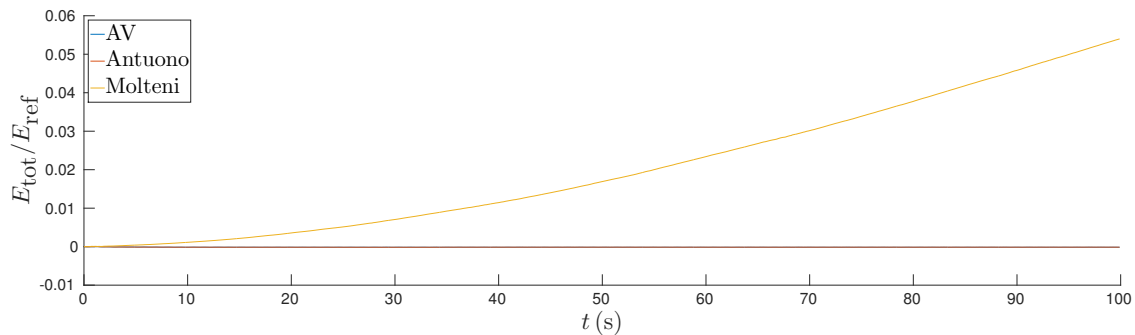


Figure 5.7: Total energy against time (s) normalised by $E_{\text{ref}} = \|\rho_0 H^3 \mathbf{g}\|$ of hydrostatic simulations using the Molteni δ -SPH, the corrected Antuono δ -SPH and the artificial viscosity cases (AV).

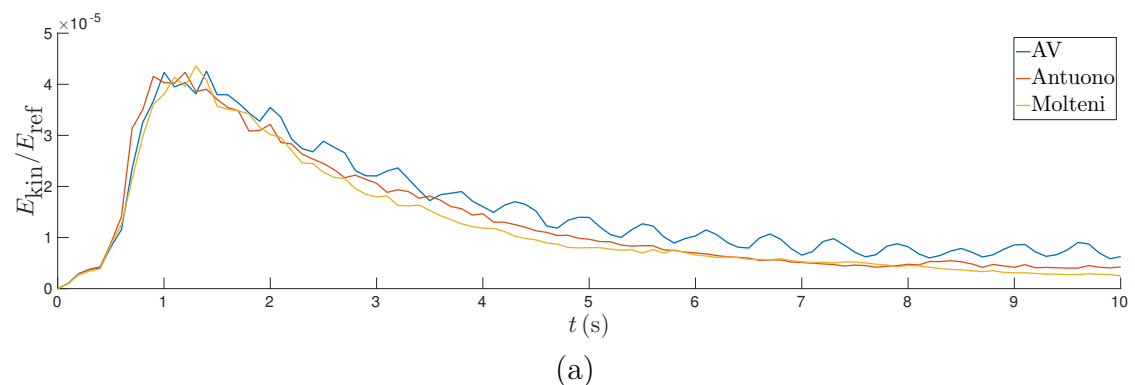
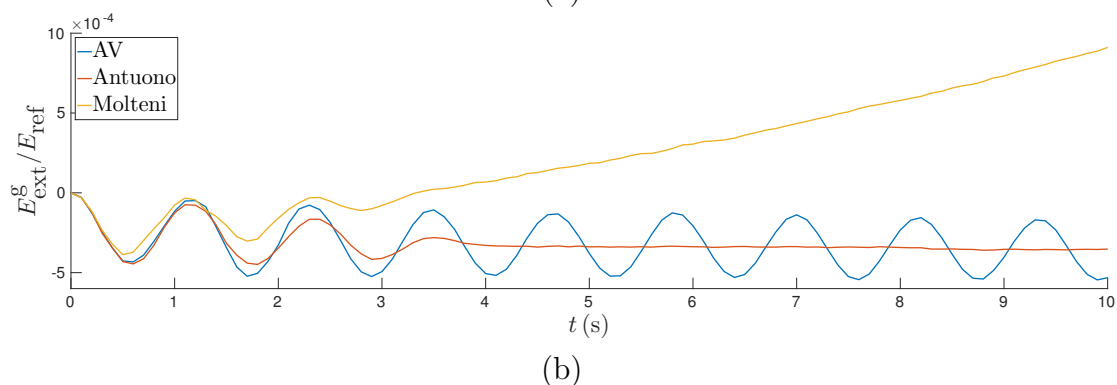
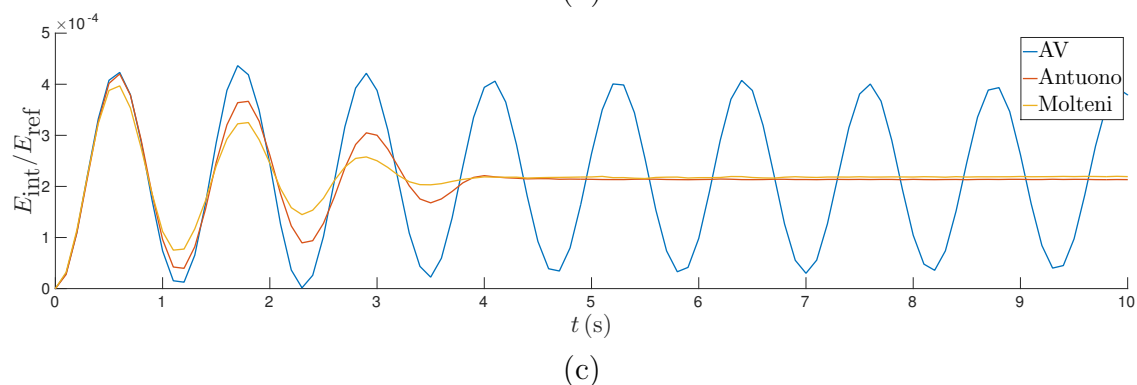


Figure 5.8: Energy components normalised by $E_{\text{ref}} = \|\rho_0 H^3 \mathbf{g}\|$ over time (s) for the first ten seconds of hydrostatic simulations using the Molteni δ -SPH, the corrected δ -SPH and the artificial viscosity (AV): (a) kinetic energy; (b) potential energy; (c) internal energy.

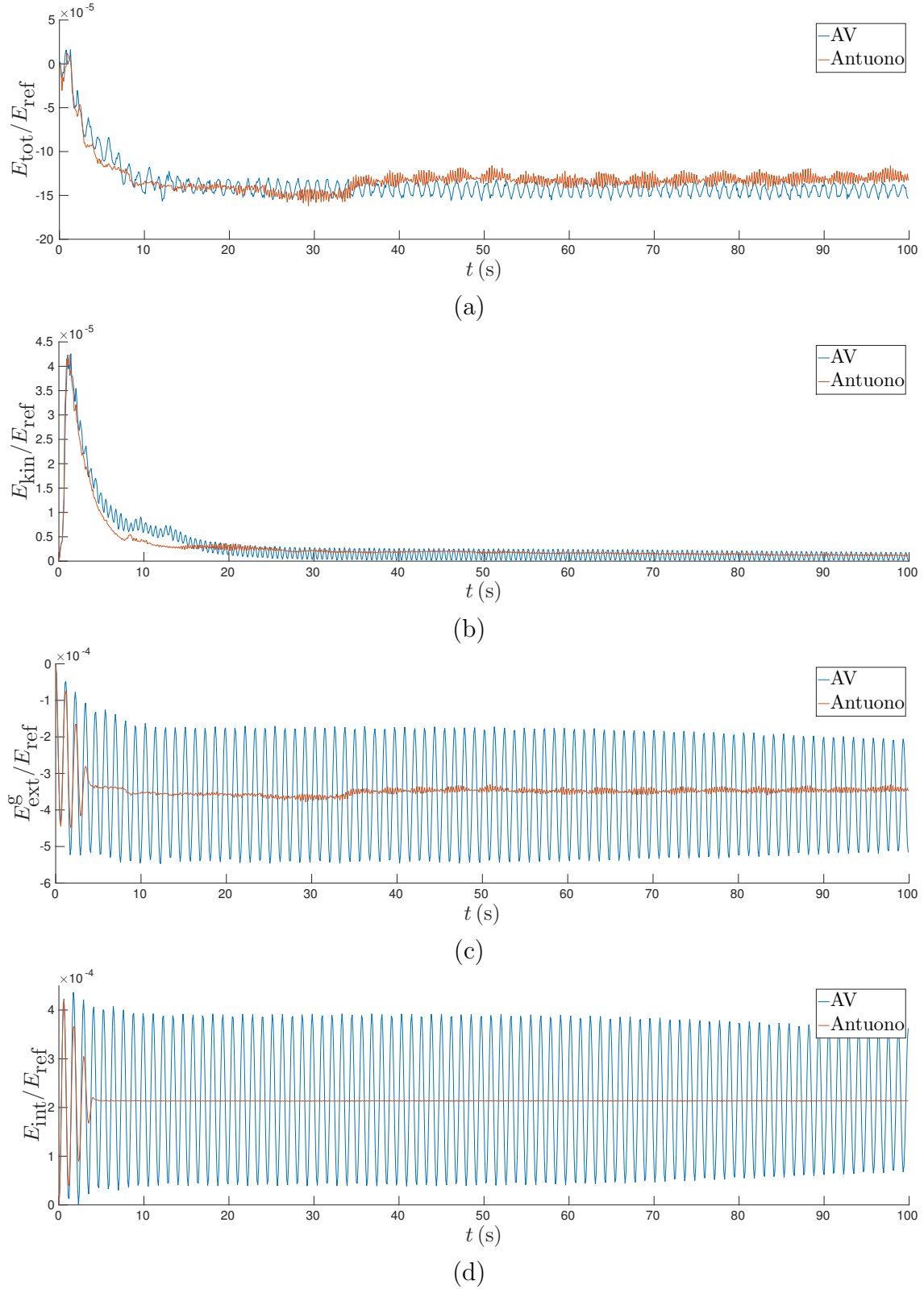


Figure 5.9: Energy components normalised by $E_{\text{ref}} = \|\rho_0 H^3 \mathbf{g}\|$ over time (s) of hydrostatic simulations using the corrected Antuono δ -SPH and artificial viscosity (AV): (a) total energy; (b) kinetic energy; (c) potential energy; (d) internal energy.

5.2 Linear sloshing

To verify the whole formulation in a moving frame of reference, a linear sloshing simulation is compared against analytical values based on potential flow theory.

The test consists of a square two-dimensional tank with a width and length $L = 1\text{m}$ and filled with water to a level $H = 0.1\text{m}$.

The tank is subjected to rectilinear sinusoidal excitation in the x -axis direction, denoted a_0 . The excitation is imposed in a moving frame of reference, as defined in Section 3.6, by $\mathbf{f}_x(t) = a_0(t) = A(2\pi f_w)^2 \sin(2\pi f_w t)$. The displacement amplitude is set to $A = 0.5\text{m}$ and the frequency to $f_w = 0.3f_0 = 0.1462\text{Hz}$ where f_0 is the first natural mode obtained from the linear theory defined by

$$f_{(m-1)} = \frac{\omega_m}{2\pi}; \quad \omega_m = \sqrt{\|\mathbf{g}\| \frac{\pi m}{L} \tanh\left(\frac{\pi m}{L} H\right)} \quad (5.1)$$

with $m = 1$. The simulation is run for a duration of six sloshing periods, $T_w = 1/f_w$, which equates to 41.05 seconds. The geometry and duration are chosen to investigate the ability of the scheme to cope with a low fill ratio and a long duration, which is a matter of interest for the experiments simulated in Section 6. The excitation parameters are chosen so that a noticeable change in fluid height and sloshing force occur without the need for simulations with millions of particles. However, such values are stretching the linear sloshing assumption.

Following reference [43], the analytic solutions for rectilinear sinusoidal excitation in the x -axis direction of the wave elevation $\zeta(x, t)$ and sloshing force $F_x(t)$ are given by

$$\zeta(x, t) = \sum_m^{\infty} \beta_m(t) \cos\left(\frac{m\pi}{L} x\right) \quad (5.2)$$

$$F_x(t) = \rho H L a_0(t) + \rho H L^2 \sum_m^{\infty} \dot{\beta}(t) \frac{1 + (-1)^{m+1}}{m^2} \quad (5.3)$$

where $\beta(t)$ and $\dot{\beta}(t)$ are obtained by solving the modal equation

$$\ddot{\beta}(t)_m + \omega_m^2 \beta(t)_m = -a_0(t) \left(\frac{2}{\pi m}\right) \tanh\left(\frac{\pi m}{L} H\right) \left[(-1)^m - 1\right]. \quad (5.4)$$

For purely lateral tank motion, the modal equation only has odd modes and the even modes are zero. Analytic solutions to equation (5.4) exist in some cases,

but in general, solutions are found with a numerical method like the Runge-Kutta scheme. In this case, the fourth-order Runge-Kutta scheme [114] is used to solve the equation for the first 25 modes, which should provide a sufficiently accurate solution to equation (5.3).

The simulation parameters are $\rho_0 = 1000 \text{ kg/m}^3$, $\gamma = 7$, and $c_0 = 10v_{\max}$ where $v_{\max} = \sqrt{|gH|}$ and $g = -9.81 \text{ m/s}^2$ is the acceleration due to gravity. Particle spacing is set to $H/\Delta x = 160$ generating 256 000 fluid particles. A free-slip boundary condition was applied using only δ -SPH for numerical dissipation with $\delta = 0.1$.

Figures 5.10 and 5.11 show comparisons of the left and right wave heights, and of the sloshing force respectively. Both the wave heights and forces are in good agreement, but there is a noticeable increasing difference between the analytical and SPH results with time. A likely explanation is that the simulation shows a small non-linear behaviour as the assumptions of the linear sloshing are close to their limit for the value of amplitude used.

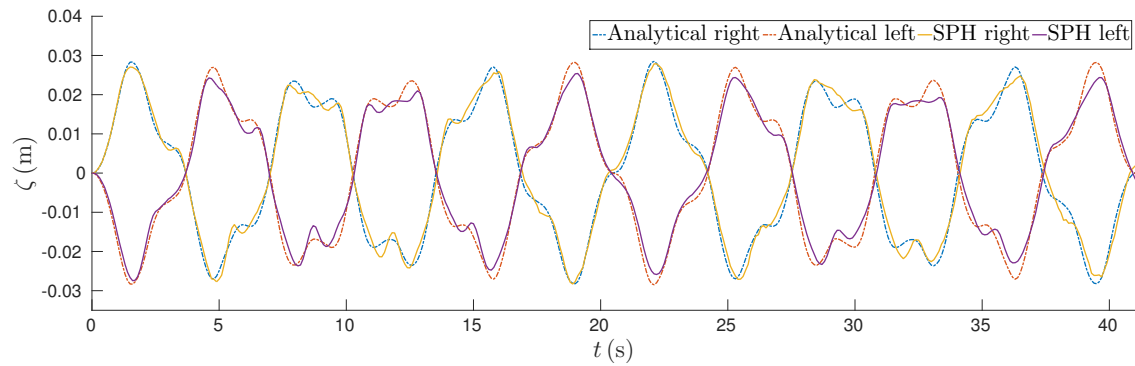


Figure 5.10: Linear sloshing wave elevation at the walls ζ (m) over time (s) for the SPH simulation and the analytical solution.

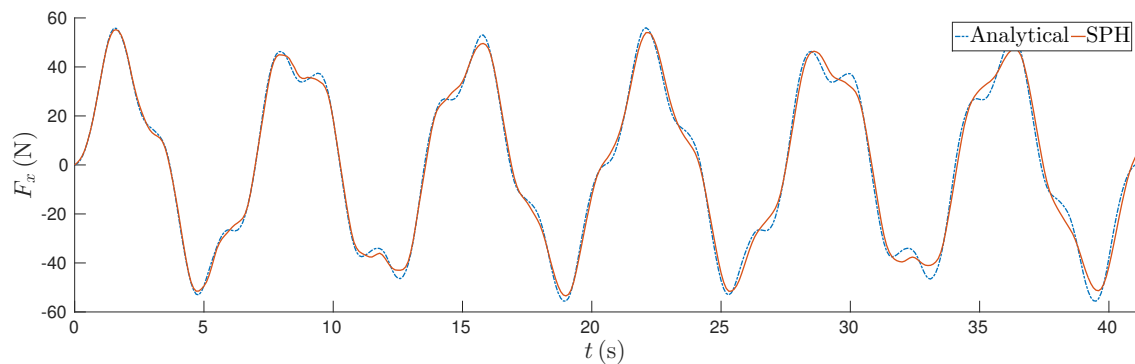


Figure 5.11: Linear sloshing fluid force in x -axis direction, (N) over time (s) for the SPH simulation and the analytical solution.

For the convergence study, the simulation was run for one second with values $h_w/\Delta x$ of 20, 40, 80 and 160, resulting in 4 000, 16 000, 64 000 and 256 000 fluid particles respectively. The accumulated error was evaluated by integrating the L_2 norm of the difference between the analytical and computed forces. The error was plotted along with the calculated order of convergence (EOC) in Fig. 5.12 where the EOC is represented by the slope of each segment, calculated as

$$\text{EOC} = \frac{\log \frac{\varepsilon(\Delta x_n)}{\varepsilon(\Delta x_{n+1})}}{\log \frac{\Delta x_n}{\Delta x_{n+1}}}$$

where $\varepsilon(\Delta x_n)$ denotes the accumulated force error using a particle spacing $\Delta x_n = \frac{1}{20} \frac{h_w}{2^n}$ for $n = 0, 1, 2, 3$. The method converges, but the EOC varies from case to case due to small oscillations in the SPH forces. Since the EOC is quite low, a further increase in resolution might not improve the accuracy significantly enough to justify the additional computational cost.

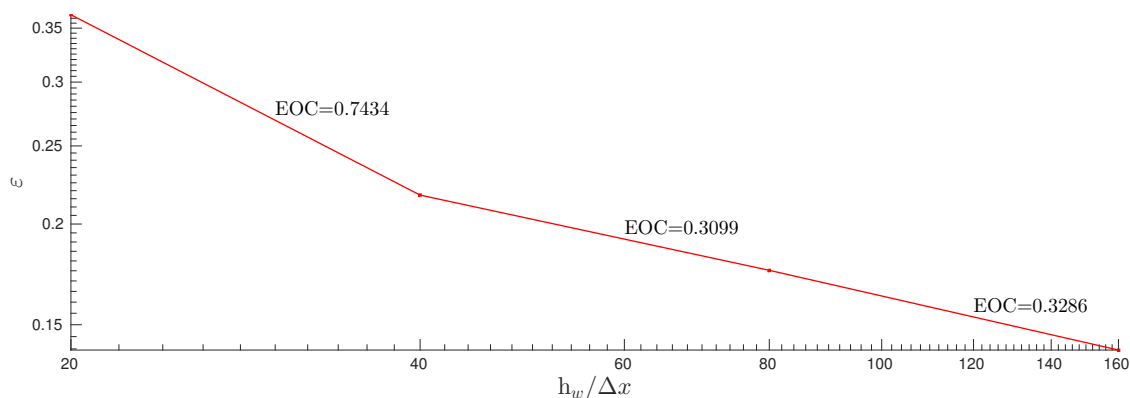


Figure 5.12: Accumulated error of the fluid force in x -axis direction in a linear sloshing simulation using different resolutions. The errors are calculated by the integral of the L_2 norm of the difference between the simulated and analytical force. The experimental order of convergence (EOC) is also shown.

5.3 Dam break

A simple two-dimensional dam break simulation is chosen as a validation benchmark. This test case is commonly studied in the SPH literature [3, 7, 90]. Here the fixed ghost particles boundary condition (GBPC) implementation in DualSPHysicsGPM is verified against published results [3] and experimental data [16].

The problem consists of an open top two-dimensional tank of width $L_w = 5.336H$

with a fluid initially occupying a domain of width $L = 2H$ and height $H = 600\text{mm}$ at the lower corner of the tank. Upon the start of the simulations, the fluid column is allowed to collapse and then plunges towards the far wall where it impacts the wall and collapses into the fluid as an overturning wave.

Following reference [3], the fluid properties are normalised by density $\rho = 1\,000\text{kg/m}^3$, gravity $\mathbf{g} = (0, -9.81)\text{m/s}^2$ and height H . This gives non-dimensional values of density $\rho_0 = 1$ and of gravity $g = 1$. Time and pressure are normalised by $t_{\text{ref}} = \sqrt{\|H/\mathbf{g}\|}$ and $P_{\text{ref}} = \|\rho\mathbf{g}H\|$, respectively. The numerical values are $\gamma = 7$, $c_0 = 10v_{\text{max}}$ with $v_{\text{max}} = \sqrt{g(H/H)} = 1$, and using artificial viscosity (3.61) with a value of $\alpha = 0.2$. Particle spacing is set to $H/\Delta x = 100$ resulting in 20 000 fluid particles, and the inter-particle pressure formulation given in equations (3.38) and (3.42) is used. It is worth to note that, in this author's opinion, the artificial viscosity value is far higher than needed and could slightly influence the fluid behaviour, mostly after the impact with the wall. However, in order to match the published results it is left at this value. The simulation using DualSPHysicsGPM is initialised from a hydrostatic pressure profile, while in reference [3] a warm-up stage was used (see Section 3.5.4). Both these initialisation methods cause travelling pressure waves throughout the simulation, although using a simple hydrostatic initialisation results in higher frequency oscillations [3]. An illustration of the simulation's set up and parameters is provided in Figure 5.13.

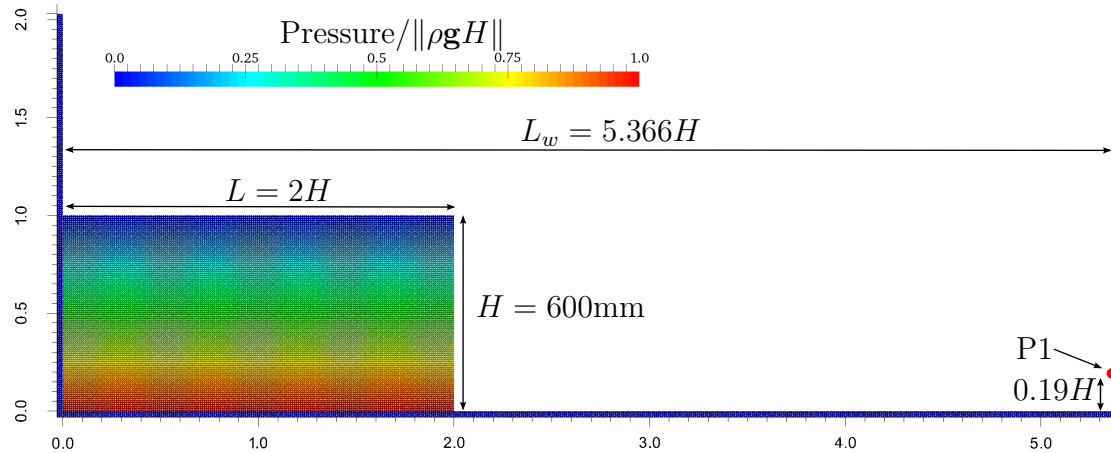


Figure 5.13: Schematic of the dimensions and initialisation of the dam break simulation.

Figure 5.14 shows the pressure contour plots obtained with particle spacing of $\Delta x = H/100$, accounting for 20 000 fluid particles. The results show very strong resemblance to the published figures. Smaller details, such as the shape of the fluid's

free-surface deformation also show strong similarities. A clear difference can be observed towards the end of the simulation, as seen at $t/t_{\text{ref}} = 7.4$ in Figure 5.14. At this time a small void is visible between the fluid and the boundary in the published results, which is caused by splashing of detached fluid particles falling onto the top of the wave. In the simulation using DualSPHysicsGPM fluid particles that travel too far above the boundary height get removed from the simulation and hence these particles do not return down to impact the wave.

Pressure at a point P1, $y = 0.19H$, is then compared against the published results and an experiment [16]. In the experiment, the pressure probe location is slightly different and it has a wide diameter rather than being at a single point, but, following references [3] and [54], this location is used for validation purposes. While there are obvious areas of agreement, such as the rise of the value of the pressure after impacting the wall, there are also clear deviations from the experimental results. Both the simulated and published results have a delay in the initial rise of the pressure at around $t/t_{\text{ref}} = 2.5$ and in the timing and amplitude of the pressure peak after the wave collapse at around $t/t_{\text{ref}} = 6$. According to reference [90], the timing of the later could be related to a single fluid formulation not being able to capture the air cushion effect. In addition to these discrepancies with the experimental data, far higher frequency oscillations are observed in the DualSPHysicsGPM simulation. These are very likely to be caused by the different initialisation techniques [3], as mentioned before.

Lastly, to validate the Antuono δ -SPH implementation, the same case was run with no artificial viscosity $\alpha = 0$ and using the Antuono scheme with $\delta = 0.1$. The results for the pressure gauge at P1 are compared against data extracted from the published results by Antuono et al. [7] and given in Figure 5.16. The exact configuration of the simulation is significantly different, from the boundary and initial conditions to the number of particles used. For the full description the reader is referred to reference [7], where the Antuono δ -SPH scheme [8] is compared against the scheme of Ferrari [47]. The results are only to show quantitative and not qualitative agreement with the published results and the ability to run the simulation without the use of additional artificial viscosity.

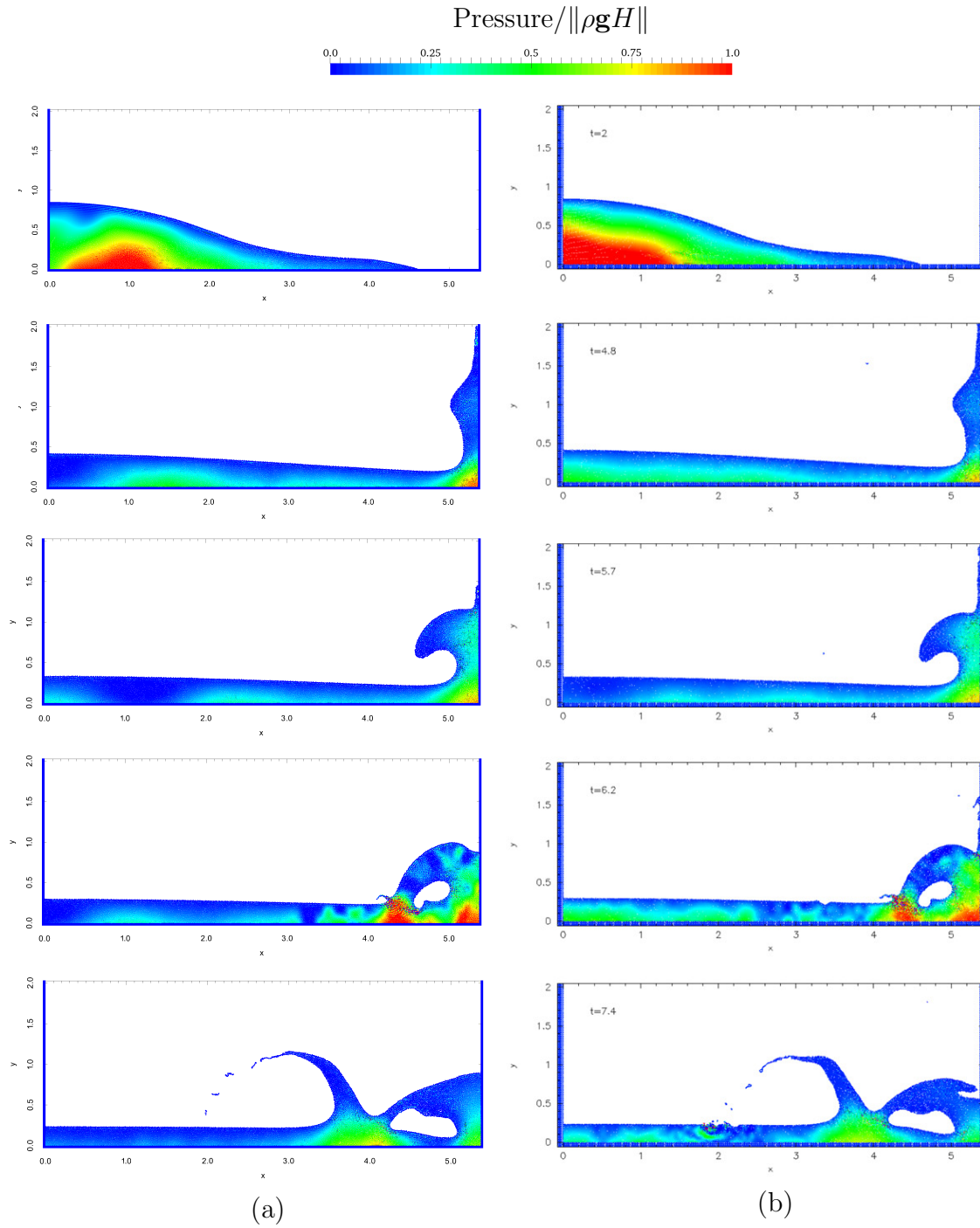


Figure 5.14: Snapshots of the pressure field in a dam break simulation, normalised by $P_{\text{ref}} = \|\rho \mathbf{g} H\|$, using the fixed ghost particle boundary condition, $H/\Delta x = 100$ and $\alpha = 0.2$. The times, normalised by $t_{\text{ref}} = \sqrt{\|H/\mathbf{g}\|}$, from top to bottom are 2, 4.8, 5.7, 6.2, 7.4: (a) DualSPHysicsGPM simulation; (b) results from [3].

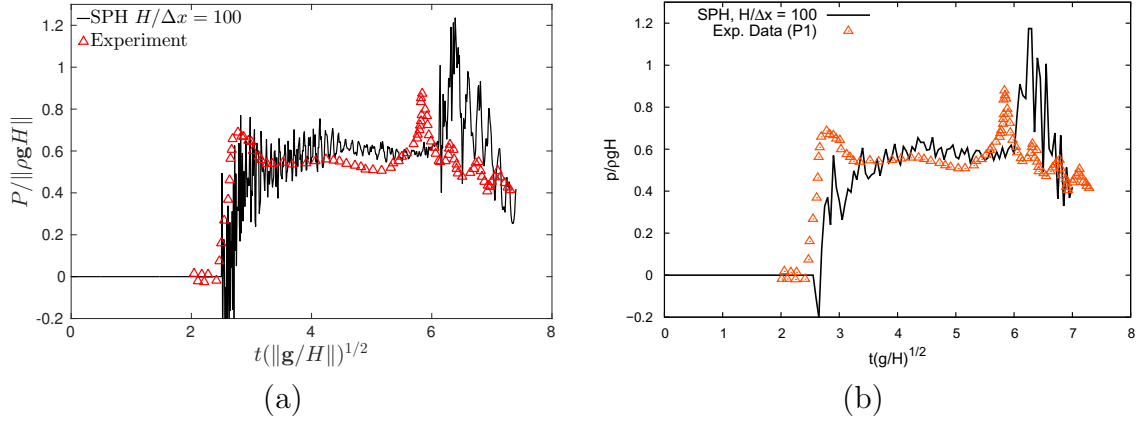


Figure 5.15: Pressure $P/(\rho g H)$, over time $t\sqrt{\|g/H\|}$ at the location of the pressure gauge P1. Comparison of the interpolated pressure from the SPH dam break simulation against experimental data taken from [3]: (a) SPH simulation using DualSPHysicsGPM; (b) the same case taken from Adami et al. [3].

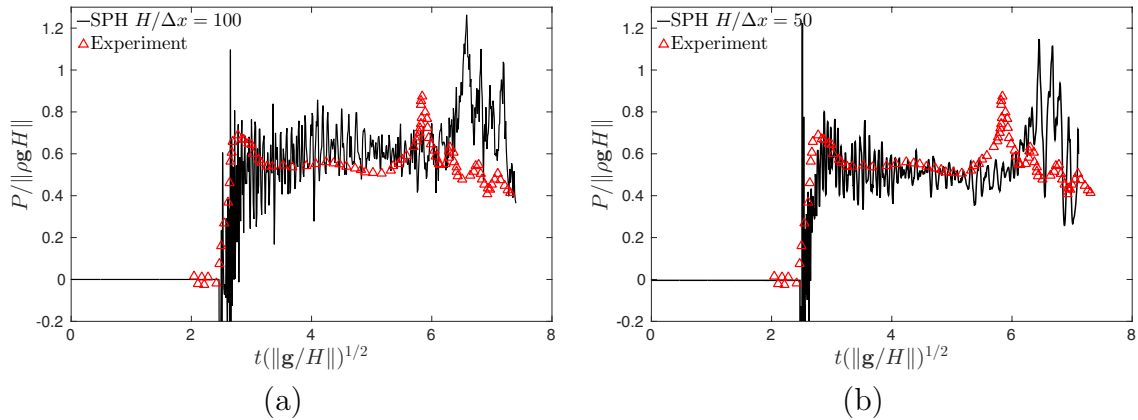


Figure 5.16: Pressure $P/(\rho g H)$, over time $t\sqrt{g/H}$ at the location of the pressure gauge P1. Comparison of the interpolated pressure from the SPH dam break simulation against experimental data taken from [3]: (a) DualSPHysicsGPM with $H/\Delta x = 100$, $\alpha = 0$ and $\delta = 0.1$; (b) extracted from data published in Antuono et al. 2012 [7] using $H/\Delta x = 50$, $\alpha = 0$ and $\delta = 0.1$.

5.4 3D dam break - SPHERIC Test 2

To test the three-dimensional capabilities of DualSPHysicsGPM the SPHERIC Test 2 [70] was chosen. This test is a simple model of green water flow on the deck of a ship, and one of the most popular tests for the validation and benchmarking of SPH codes. The experiments were carried out by the Maritime Research Institute Netherlands (MARIN) and the description and data are openly available at the SPHERIC web page [136].

This experiment takes place in an open roof tank where a compartment is filled with water enclosed behind a door. At the start of the experiment, the door is removed and the water flows into the tank where it hits a small box placed in the fluid domain before impacting the far wall. Experimental measurements of water heights and pressure values at specific points are taken to be compared against simulations. The geometry of the computational domain, its dimensions, and the position of the water height probes and pressure gauges are shown in Figure 5.17.

Two cases are simulated, each with a different choice of parameters. The first case uses the DualSPHysics with the dynamic boundary condition (DBC), the direct density formulation (3.40–3.41) and both artificial viscosity and the Molteni δ -SPH scheme with $\alpha = 0.05$ and $\delta = 0.1$, respectively. These settings are chosen to be as similar as possible to those of reference [38] where the DBC is compared against other boundary conditions. It is noted however that the δ -SPH scheme used in reference [38] is that of reference [44] which was included by the main developers for internal testing only and was never made available to other developers or users. The second case is using what is found to be the best current parameters for DualSPHysicsGPM. This includes the continuity formulation (3.38–3.39) with $\alpha = 0.02$ and $\delta = 0.1$ for artificial viscosity and δ -SPH respectively. Both simulations have a particle spacing of $\Delta x = 0.01\text{m}$ giving rise to 652 212 and 676 500 fluid particles and 119 400 and 379 048 boundary particles for the DBC and GPBC cases, respectively. The fluid properties are those of water, with density $\rho = 1000\text{kg/m}^3$ and $g = -9.81\text{m/s}^2$ the acceleration due to Earth's gravity. The constitutive equation has the exponent $\gamma = 7$ and the numerical speed of sound $c_0 = 20\sqrt{|gH|}$ where $H = 0.55\text{m}$ is the initial fluid height.

The wave heights of both simulations agree reasonably well with the experimental data. Varying accuracy is observed at different stages of the simulations shown in Figure 5.18. While qualitatively the simulations are hard to distinguish, there are

a few noticeable differences. The first is the early spike of the GPBC simulation in position H1, which is located just after the small box in the fluid domain and the spike of the DBC in position H2 which is located just before the box which occurs around 0.6 seconds into the simulation. These spikes are caused by the shape of the fluid splashes when impacting the box - in the GPBC case the splashing surges forward while in the DBC case it is more vertical and curls slightly backwards. A further noticeable difference is the capturing of the returning wave towards the far wall around 6 seconds, seen in the measurements at position H1, which is far better captured with the GPBC. Overall, the timings occur slightly earlier and in better agreement with the experiment in the GPBC case, which is likely due to the lower value of artificial viscosity.

The pressures shown in Figure 5.19 exhibit a similar trend, with both simulations agreeing reasonably well with the experimental data. However, there is a significant difference in the location of the points where the pressure interpolation takes place. This issue is illustrated in Figure 5.21 where the pressure interpolation points around the box are shown as the fluid impacts the box. Having the interpolation points further away from the boundary is likely the reason the pressure readings in the DBC case are noticeably lower than the GPBC and experimental ones. However there is a minor drawback in having the pressure interpolation points exactly on the box: it becomes more sensitive to the splashing and can result in gaps in the readings as observed in the plot for location P7 in Figure 5.19. Similarly to the wave heights measurements, the timing is slightly better captured in the GPBC simulation, likely due to a combination of the lower artificial viscosity and the location of the pressure interpolation points. However, the lower numerical dissipation also leads to a small increase in oscillations of the pressure values.

This test case validates both the fixed ghost particles boundary condition and the corrected Antuono δ -SPH implementation in DualSPHysicsGPM in three dimensions. While both the DBC and the GPBC simulations are capable of reproducing the experimental results to a similar degree of accuracy, there are two main advantages of using the later. The first is that using the fixed ghost particles boundary condition eliminates the non-physical gap between the fluid and boundary particles. The second is the ability to use less artificial dissipation to obtain results of a similar quality. However, it is important to emphasise that the computational time for the GPBC case is far larger than the DBC case, taking approximately ten times as long. The extra computational time is mostly due to the use of the Antuono δ -SPH

scheme, the use of double precision in the GPBC case, and the additional boundary particles which number around 200 000. As the duration of this test is very short and highly dynamic, the Molteni [96] or other second order δ -SPH schemes (see Section 3.5.3), which are far cheaper than the fourth order Antuono scheme, might be more suitable for this case. For a study of different numerical dissipation terms for highly dynamic short duration cases the reader is referred to reference [7].

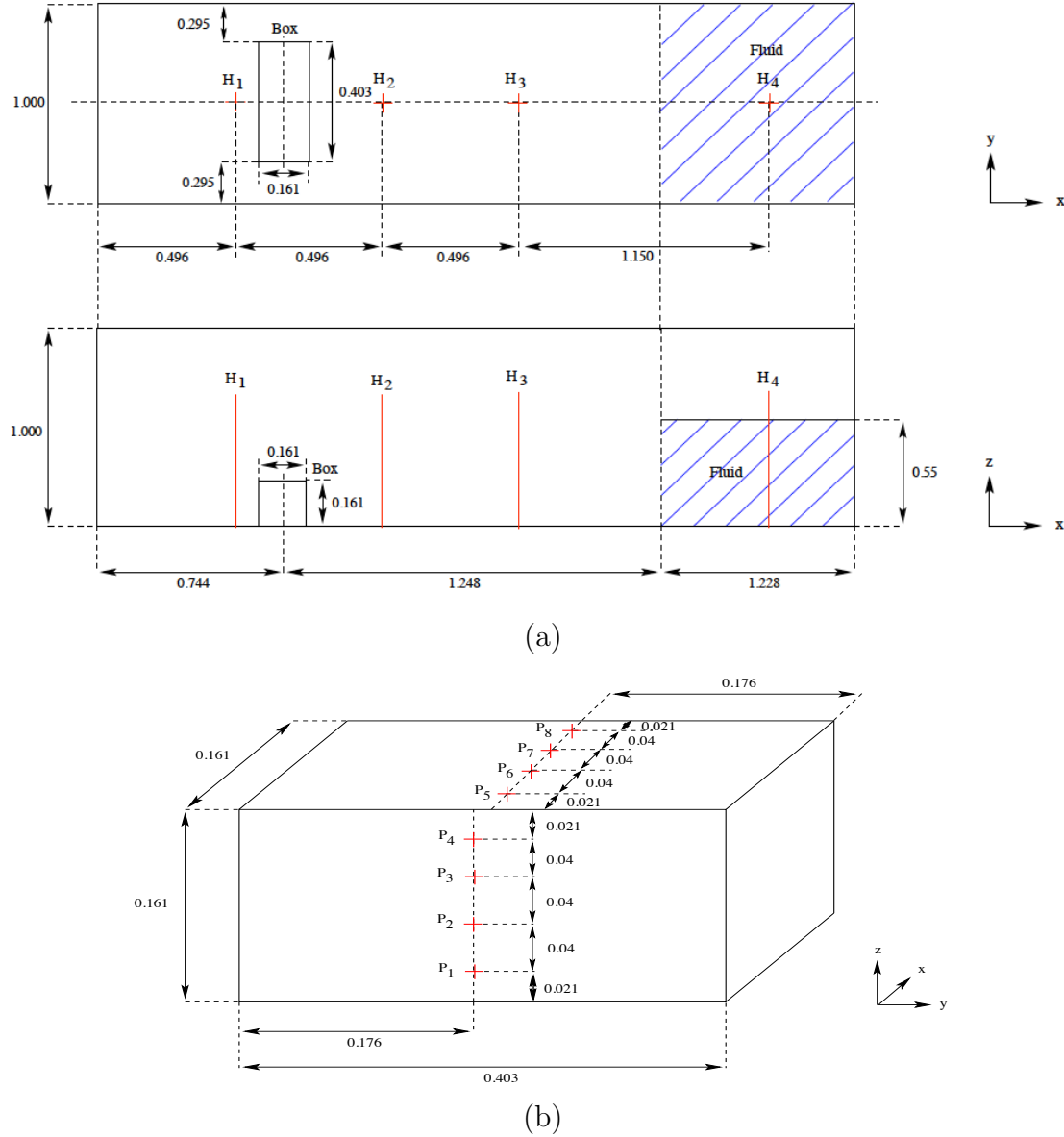


Figure 5.17: Schematics of the three-dimensional dam break SPHERIC validation test case – SPHERIC Test 2: (a) geometry of the computational domain and location of vertical wave probes; (b) location of pressure gauges on the box. Taken from reference [70].

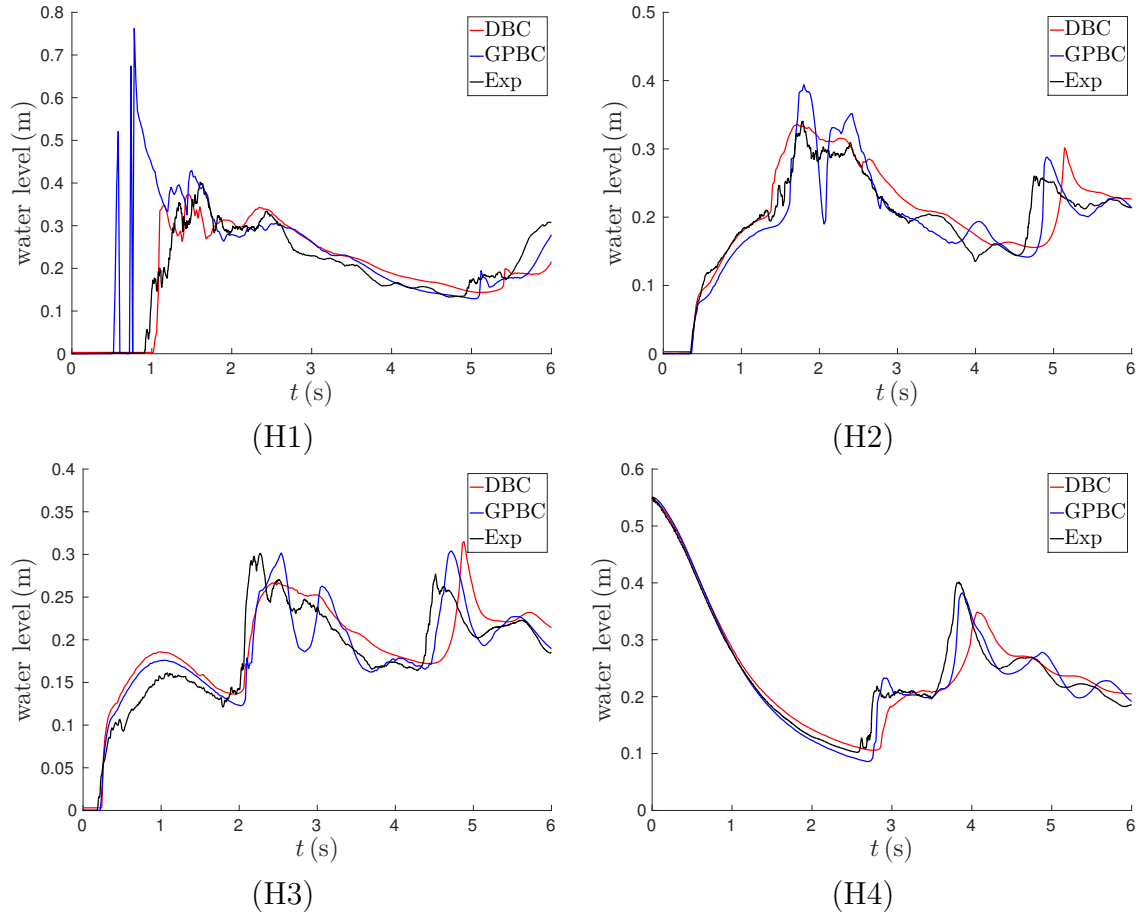


Figure 5.18: Comparison of the water heights (m) over time (s) calculated by DualSPHysicsGPM (GPBC) and DualSPHysics (DBC) against the experimental data for SPHERIC Test 2. The labels in brackets correspond the height probes shown in Figure 5.17.

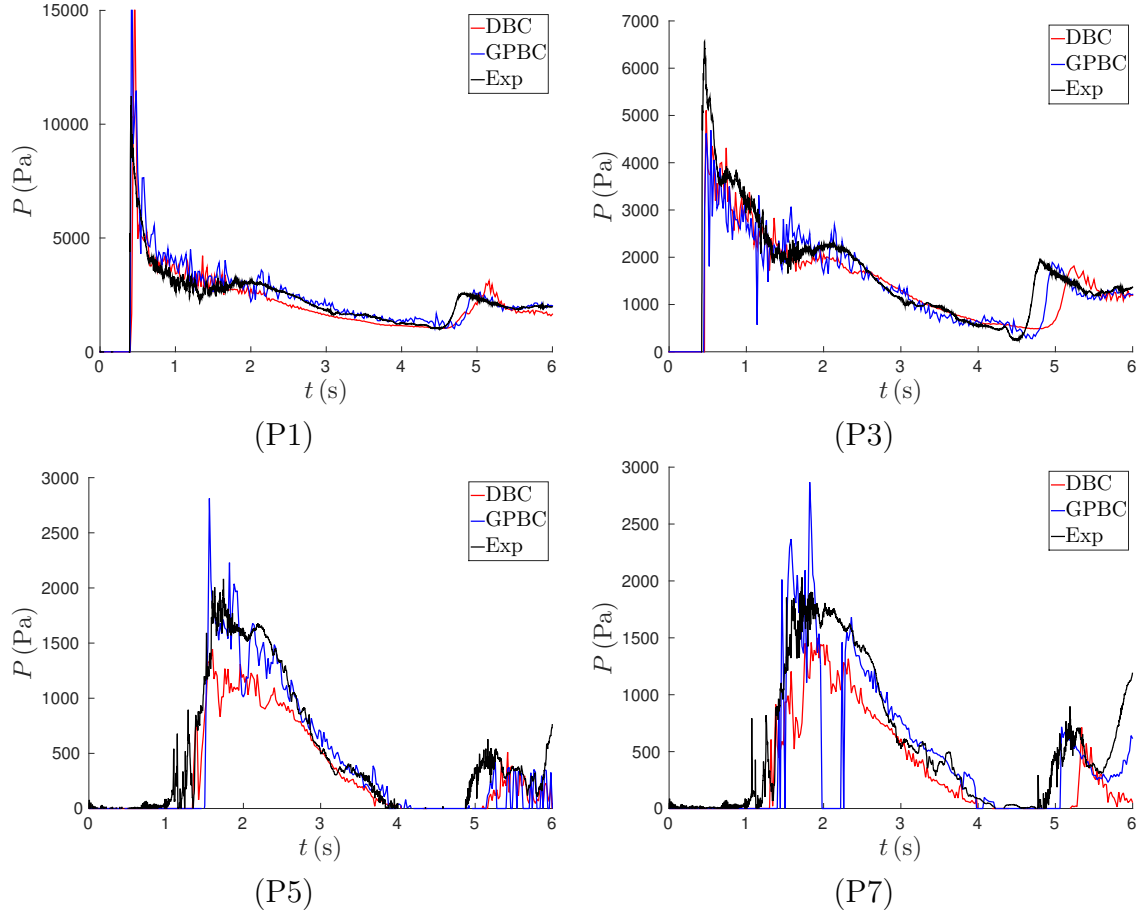


Figure 5.19: Comparison of the interpolated pressure (Pa) over time (s) calculated by DualSPHysicsGPM (GPBC) and the DualSPHysics (DBC) against the experimental data for SPHERIC Test 2. The labels in brackets correspond the pressure gauges shown in Figure 5.17.

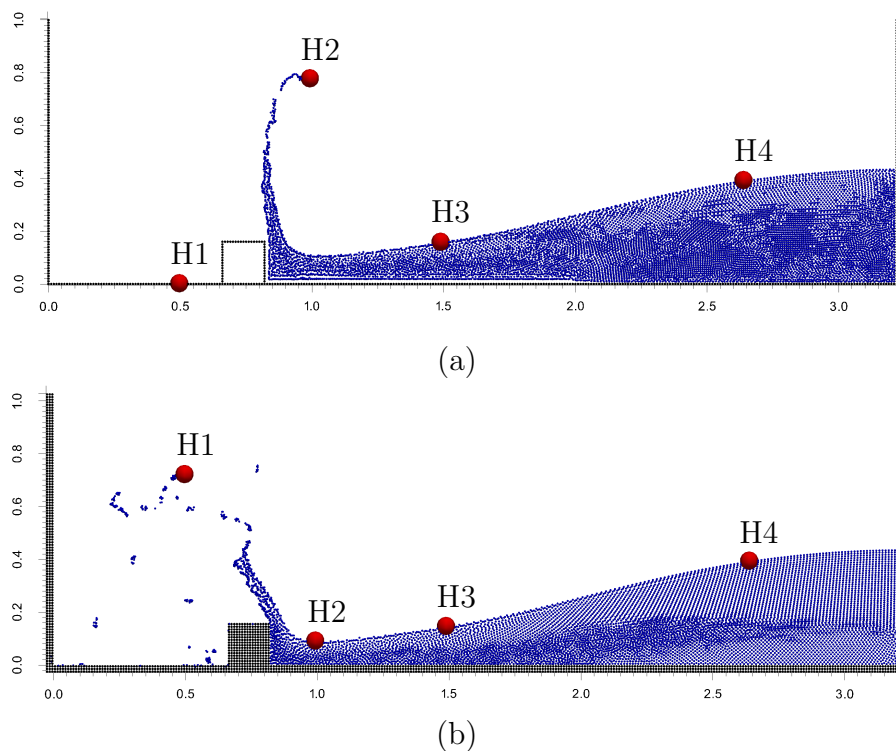


Figure 5.20: Two-dimensional slices of each case at time $t = 0.6\text{s}$ showing the interpolated heights (red circle) at the experimental wave probes' locations (a) using the dynamic boundary condition (DBC); (b) using the fixed ghost particle boundary condition (GPBC). At this time both simulation exhibit splashing which appear as spikes in the heights time-history in Figure 5.18. Also visible in (a) is the gap between the fluid and the boundary particles.

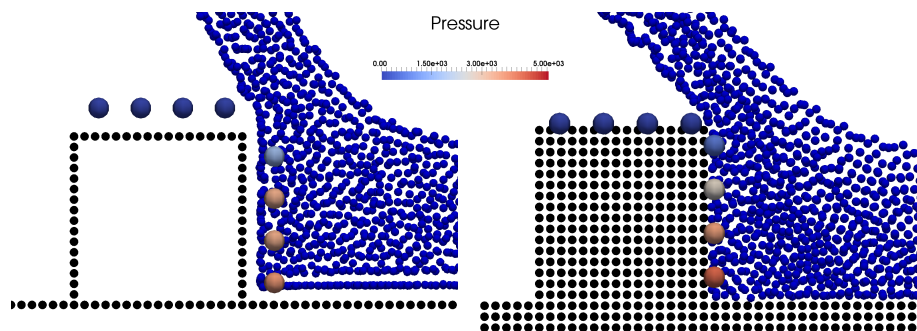


Figure 5.21: Two dimensional slice comparing the fluid structure interaction between the DBC (left) and the GPBC (right) and showing the pressure integration points for gauges P1–P8. In the DBC case a significant gap appears between the fluid and the boundary, requiring the measurements of the pressure gauges on the box to be taken at a distance of $2h$ away from the physical location.

Chapter 6

Validation against sloshing experiments

This chapter investigates the capabilities of the SPH method to simulate real world sloshing experiments. SPH simulations are compared against experiments conducted by ESA/ESTEC on the HYDRA multi-axis hydraulic shaker located in the ESTEC test centre [77].

The experiments were carried out using two different tank geometries: a rectangular tank and a pill-shaped tank. The dimensions of the rectangular tank correspond to an experiment which studies tuned liquid dampers [121] and acts as a verifications test for the simulation tools. The pill-shaped tank is representative of the actual shape of fuel tanks in spacecraft and satellites. Schematics of both these tanks are shown in Figure 6.1. The tanks were subjected to lateral harmonic excitation in a range of frequencies. Each frequency was dwelt on for a period of time ranging from 60 to 300 seconds.

The rectangular tank has a width $l = 0.59\text{m}$, a depth $d = 0.335\text{m}$ and a height $h_t = 0.25\text{m}$ and was filled with water to a height $h_w = 0.03\text{m}$. These dimensions give a first natural mode with a frequency $f_0 = 0.4578\text{Hz}$ and a *stretching ratio* of tank width to fluid height as $l/h_w = \frac{59}{3}$. All these values are summarised in Table 6.1. The quantities measured were water heights and sloshing forces. The water heights were measured using wave gauges placed at two locations: the centres of the left wall and of the tank with coordinates $(0, 0.5d)$ and $(0.5l, 0.5d)$, respectively. These locations will be denoted as H1 (centre of left wall) and H2 (tank centre). Sloshing forces were measured using tri-axial component force sensors mounted on the frame of the tank. Since these sensors measured the forces of both the fluid and the tank, forces from a dry run were subtracted from the raw measurements to give only the sloshing force.

The pill-shaped tank consists of two spherical caps of radius $h_{t1} = 0.144\text{m}$, connected with a cylinder with a height of $h_c = 0.6\text{m}$ and filled with water to a level of $h_w = 0.5\text{m}$. The first natural mode for this case corresponds to a frequency $f_0 = 1.78\text{Hz}$ and the *stretching ratio* is $2h_{t1}/h_w = 0.576$. These values are summarised in Table 6.2. Forces in all axial directions were recorded using the tri-axial component force sensors in the same manner as in the rectangular tank. However, no wave gauges were installed in the pill-shaped tank so the water heights are not compared.

The validation of the SPH methodology implemented in DualSPHysicsGPM will be described in the next sections. Section 6.1 compares the rectangular tank experiments against two-dimensional simulations. These were conducted for a choice of three frequencies where the simulation was run for a duration of 300 seconds, covering both transient and steady-state phases of the flow. Section 6.2 presents a three-dimensional simulation of the rectangular tank experiment at the natural frequency and for the steady-state phase only. Finally, section 6.3 provides a comparison of the simulation in the pill-shaped tank against the experiment, which covers the transition of the flow from rest to violent sloshing.

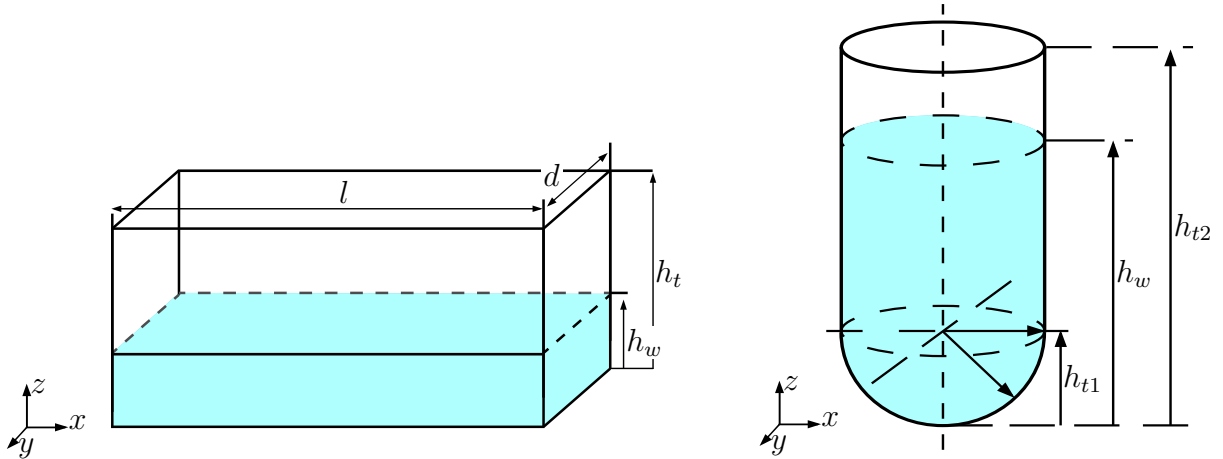


Figure 6.1: Schematics of the rectangular and pill-shaped tank used for sloshing experiments. Taken from [77].

Table 6.1: Specifications of the rectangular tank geometry.

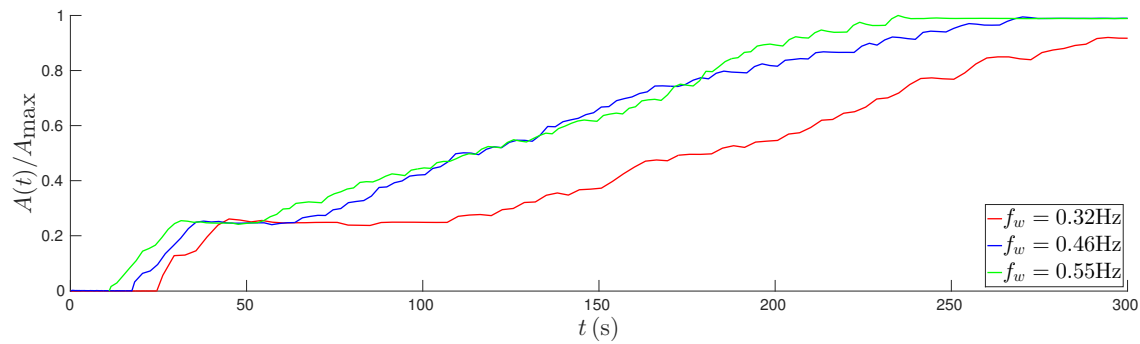
Specification	value
tank height h_t	0.25m
tank length l	0.59m
tank depth d	0.335m
water height h_w	0.03m
first natural mode f_0	0.4578Hz
stretching ratio l/h_w	19.6667

Table 6.2: Specifications of the pill-shaped tank geometry.

Specification	value
spherical cap h_{t1}	0.144m
cylindrical height h_c	0.6m
tank height $h_t = h_c + 2h_{t1}$	0.888m
water height h_w	0.5m
first natural mode f_0	1.78Hz
stretching ratio $2h_{t1}/h_w$	0.576

6.1 Rectangular tank long duration

The long duration simulations of the rectangular tank were run at three different frequencies: one below the first natural mode, one at it, and one above it. These correspond to $f_{w1} = 0.7f_0 = 0.32\text{Hz}$, $f_{w2} = f_0 = 0.46\text{Hz}$ and $f_{w3} = 1.202f_0 = 0.55\text{Hz}$ respectively. The displacement amplitude, A , for each frequency was built up during each run to a maximum value $A = 20\text{mm}$ where it dwelt for a time, as illustrated in Figure 6.2. Each run was simulated for a duration of 300 seconds using acceleration data directly from the experiment as input for the prescribed external force in the momentum conservation equation in a moving frame of reference (3.75). In the experiment, after dwelling at the maximum amplitude, the amplitude was reduced to a very small value $A = 0.005\text{mm}$ before the next frequency was run. While this significantly reduced the sloshing motion between each frequency and allowed reaching a steady state, the initial conditions did not correspond to rest. This was a significant difference with respect to the initial conditions used in the simulations that started from complete rest.

Figure 6.2: Rectangular tank normalised displacement for the frequencies $f_w = 0.32$, $f_w = 0.46\text{Hz}$ and $f_w = 0.55\text{Hz}$.

Due to the long duration of the experiment at each frequency, the simulations were attempted in two-dimensional space, using a particle spacing $\Delta x = 0.0005\text{m}$. This resolution represents 60 particles in the fluid height, i.e. $h_w/\Delta x = 60$, and gives rise to 70 800 fluid particles and 6 576 boundary particles. The fluid is water, $\rho_0 = 1,000\text{kg/m}^3$, $\gamma = 7$ and the numerical speed of sound was chosen as $c_0 = 20\sqrt{\|\mathbf{g}h_w\|}$. A laminar dynamic viscosity $\nu = 10^{-6}\text{kg/(ms)}$ is considered with a no-slip boundary condition at the walls of the tank and the corrected δ -SPH scheme with $\delta = 0.1$ applied.

Despite the fact that the rectangular shaped tank is considered a *simple* test used for the verification of the SPH simulations, it proved to be extremely challenging due to two factors: the low fill ratio of the tank and the extended durations of the experiment at each frequency.

Concerning the first of these two issues, the high stretching ratio of the liquid mass used in the experiment required a small particle spacing $\Delta x = 0.0005\text{m}$ to represent the fluid height with sufficient resolution. This caused a large variation in scale between the particle spacing and the width of the tank which required the use of double precision to reduce numerical errors, and in turn a significant increase of the computational cost.

A simulation of 300 seconds with a smoothing length $h = 1.5\Delta x = 0.00075\text{m}$ required approximately 50 million time-steps for each run and took around one week on a Nvidia Tesla K40 GPU card. This number of time-steps is a direct consequence of the time-step restriction of explicit time-stepping schemes. Since errors in SPH can persist and amplify unbound over time, such huge number of time-steps makes these simulations extremely sensitive to any spurious effects. Despite these difficulties, the proposed SPH scheme was capable of reasonably simulating the experiment, as described in detail in the following paragraphs.

The first case corresponds to a frequency $f_{w1} = 0.32\text{Hz}$ which is lower than that of the first natural mode. This case generated very low fluid motion with no wavefront and was found to be the hardest to simulate as it suffers from many of the known issues found in the hydrostatic case, making it very sensitive to numerical errors. Further, the wave elevation at the centre of the tank is so small that using only 60 particles to represent the height of the fluid means that the difference between the initial and maximum water level is less than the particle spacing. This is evident especially from the comparison during the transient phase. Despite this, the simulation still displays good agreement with the experiment towards the end of the 300

seconds period for both the water heights and force. However, it should be noted that the force is far smoother than that measured in the experiment, but with the same frequency and similar magnitude. Further, up to approximately 250 seconds, the wave heights at the centre of the tank (H2) were too low in the simulated case to be properly captured at this resolution.

The frequency of the second case is $f_{w2} = 0.46\text{Hz}$ which corresponds to the first natural mode. This case exhibits more complex dynamics: A proper wavefront formed and large splashes were observed when the wave impacted the wall, although there is no clear evidence of wave breaking. This case proved to be the simplest to simulate, with the simulation catching up with to the experiment quite early in the transient stage. After 200 seconds, the secondary waves are also reasonably captured at both locations. The force shows very similar peaks, with only some discrepancies in the shape, but is in very good overall agreement. The simulated water heights at location H1 show spikes which are far higher than those measured in the experiment. This issue requires further investigation, but it is conjectured that such discrepancy is partially due to discretisation errors in the SPH summations, associated with the fluid splashes where there are only a very few particles involved in the interaction. This is illustrated in Figure 6.12 where the height measurement in the SPH simulation at location H1 for two subsequent peaks are shown. The peak taken at $t = 293\text{s}$ vastly overestimates the fluid height due to only a few particles that broke off from the bulk of the fluid and splash against the wall. In the following peak, taken at $t = 295\text{s}$, no such particle behaviour is observed and SPH offers a much better agreement with the experimental data.

The third case with a frequency $f_{w3} = 0.55\text{Hz}$, which is above the first natural mode, displays all of the characteristics of the previous case, but with the addition of wave breakage. The results are once again in good agreement for both water heights and force. The biggest difference is that the water heights take much longer to catch up with the experiment. More specifically, the secondary waves following the wavefront only get resolved towards the end of the 300 second period. Similarly to the previous case, large spikes are observed at location H1 when the fluid splashes.

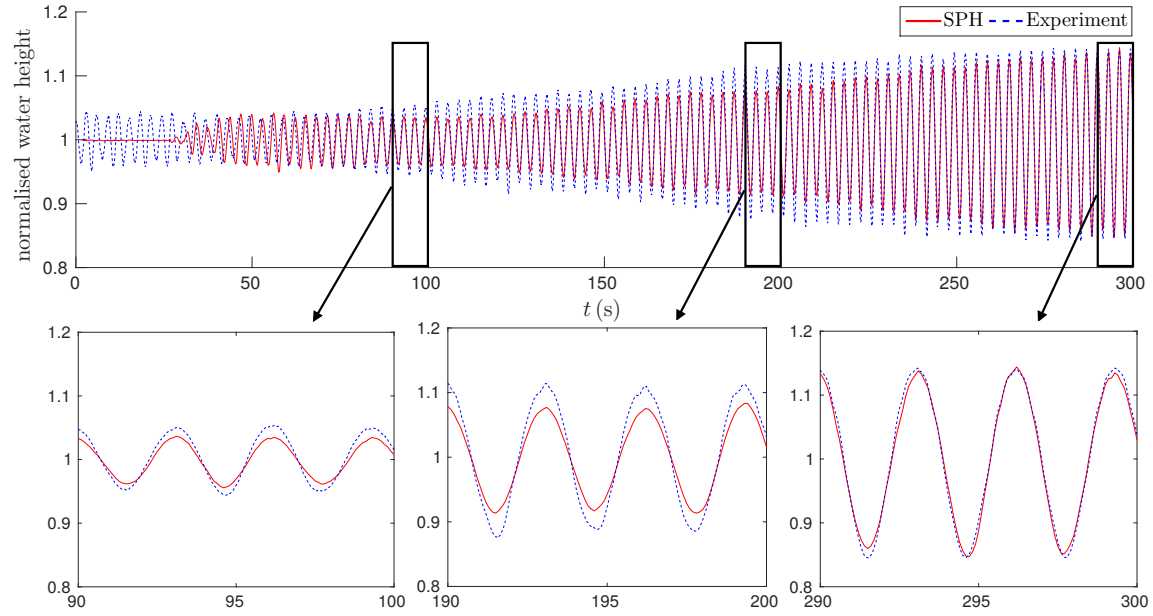


Figure 6.3: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$ at location H1.

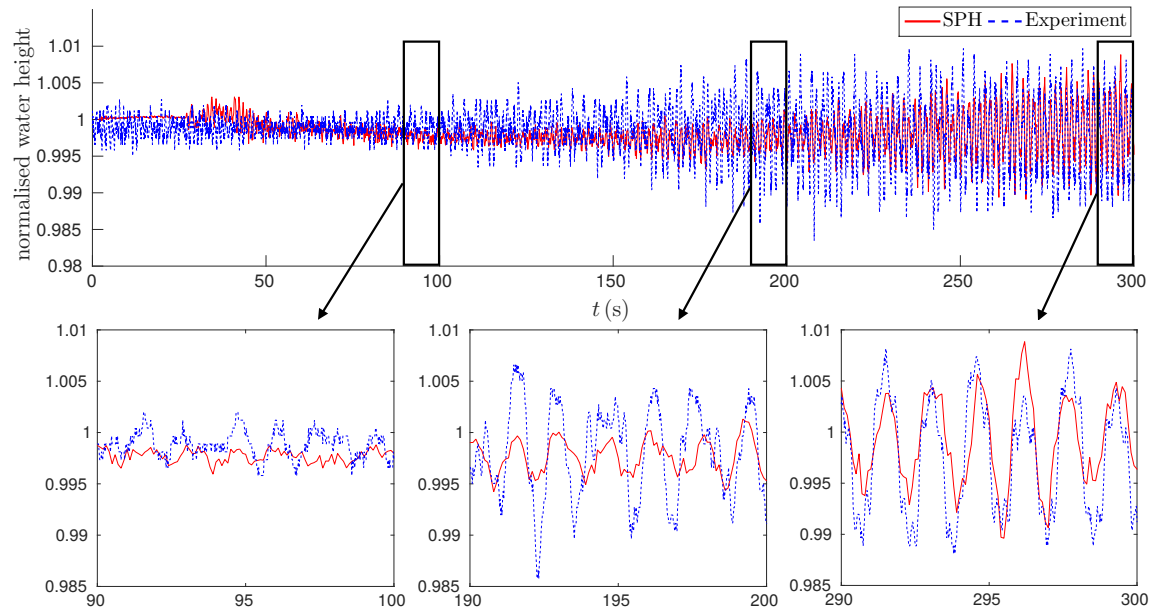


Figure 6.4: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$ at location H2.

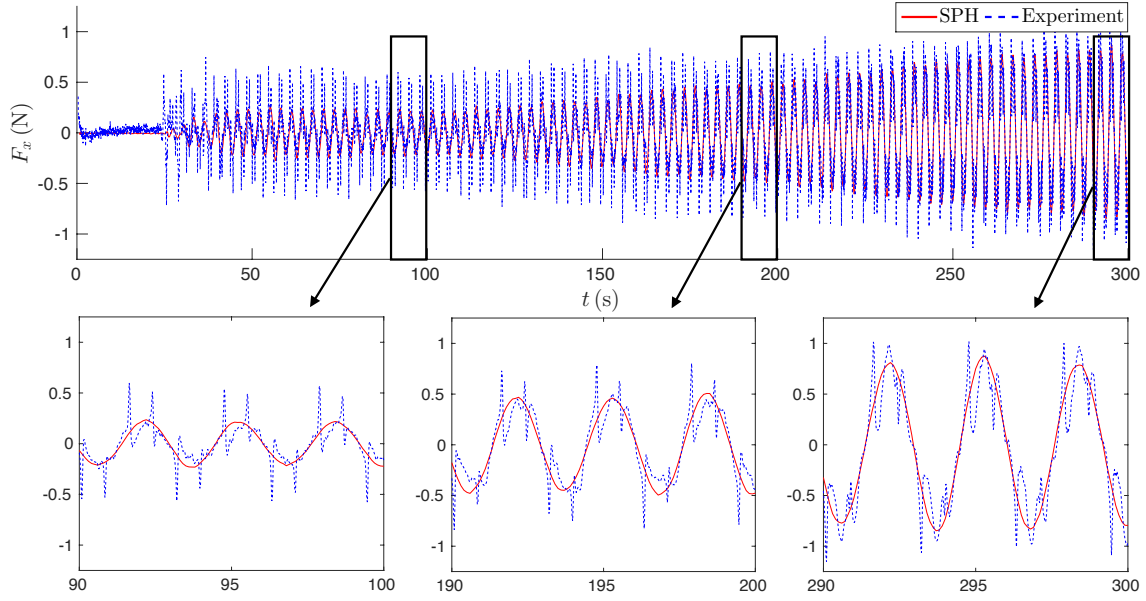


Figure 6.5: Sloshing force (N) over time (s) of the rectangular tank sloshing experiment with excitation frequency $f_{w1} = 0.32\text{Hz}$.

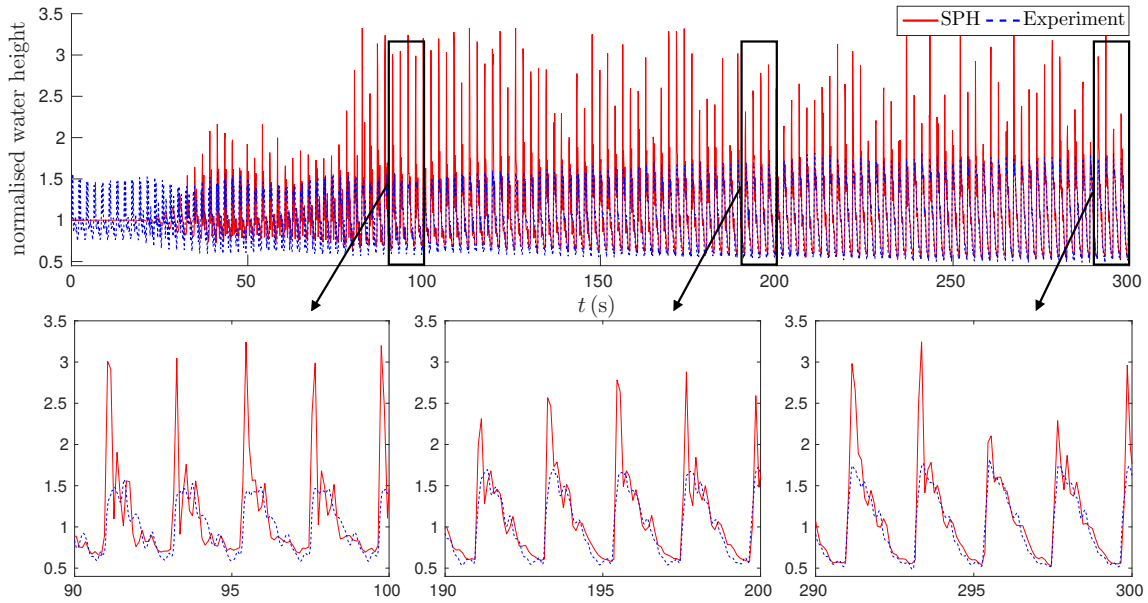


Figure 6.6: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$ at location H1.

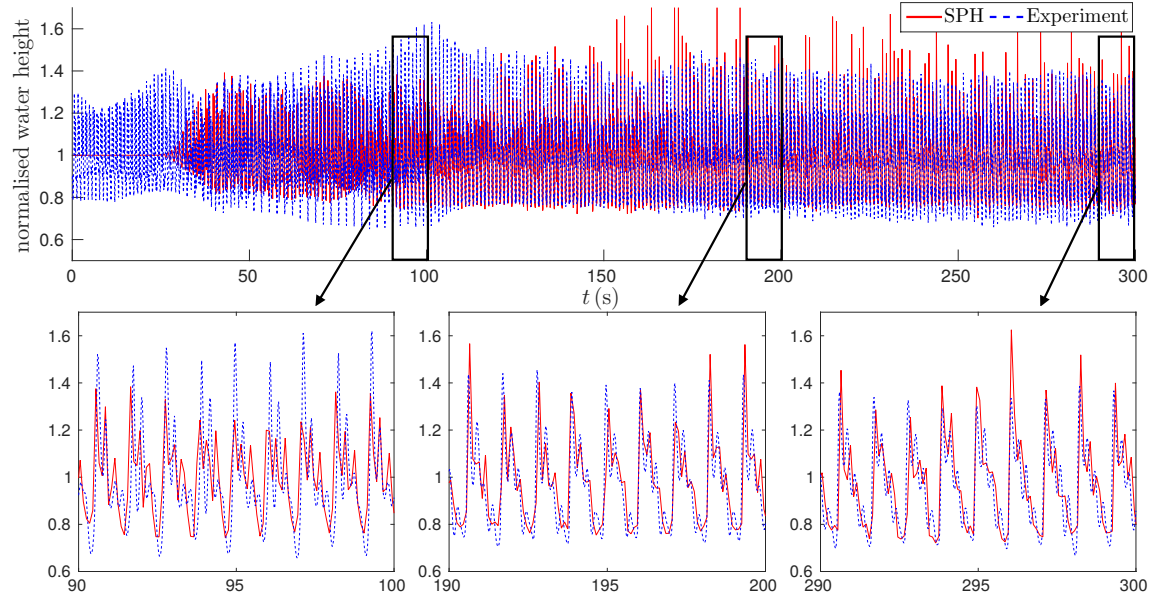


Figure 6.7: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$ at location H2.

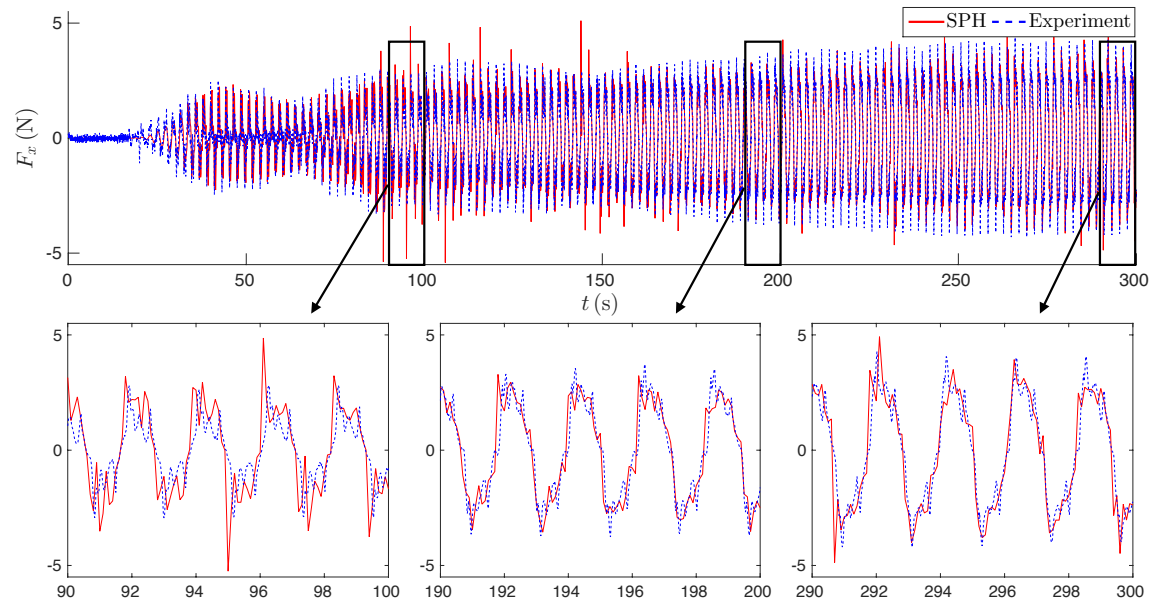


Figure 6.8: Sloshing force (N) over time (s) of the rectangular tank sloshing experiment with excitation frequency $f_{w2} = 0.46\text{Hz}$.

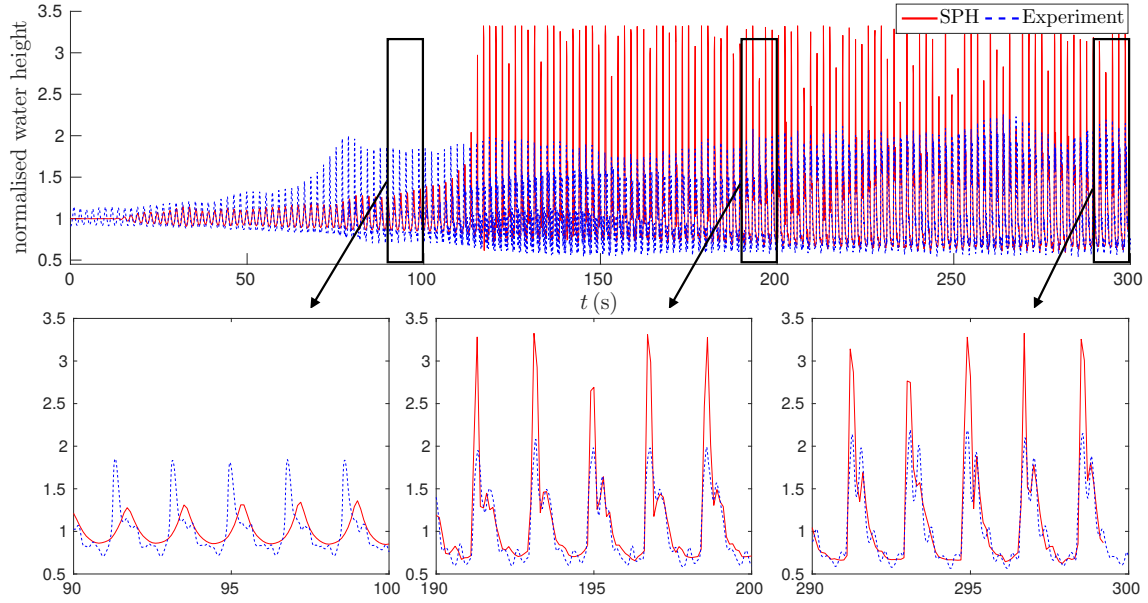


Figure 6.9: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$ at location H1.

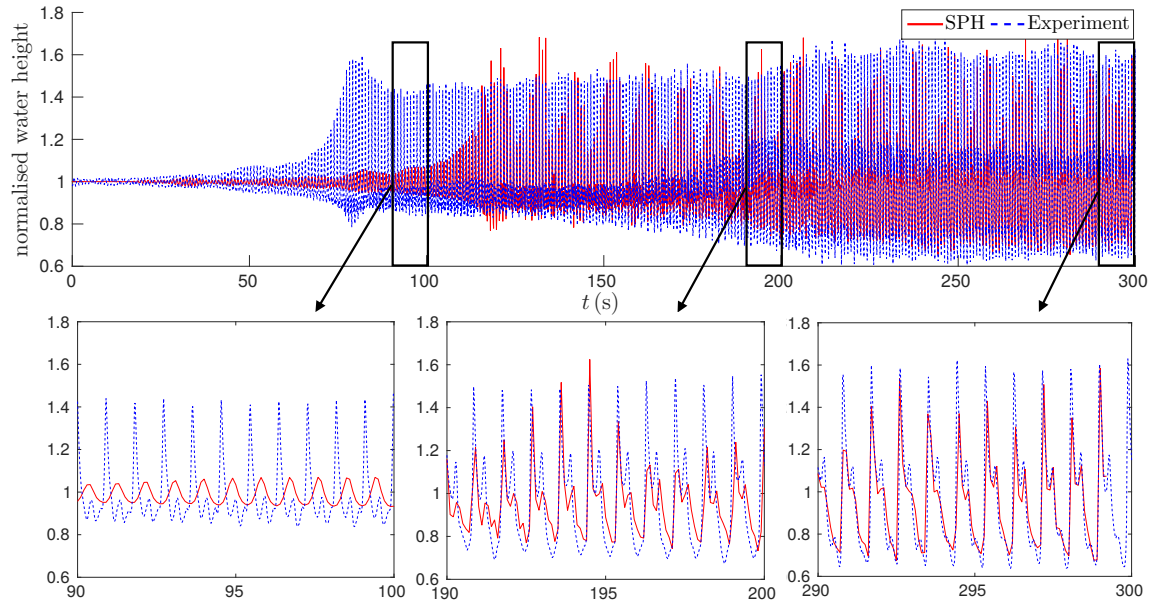


Figure 6.10: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$ at location H2.

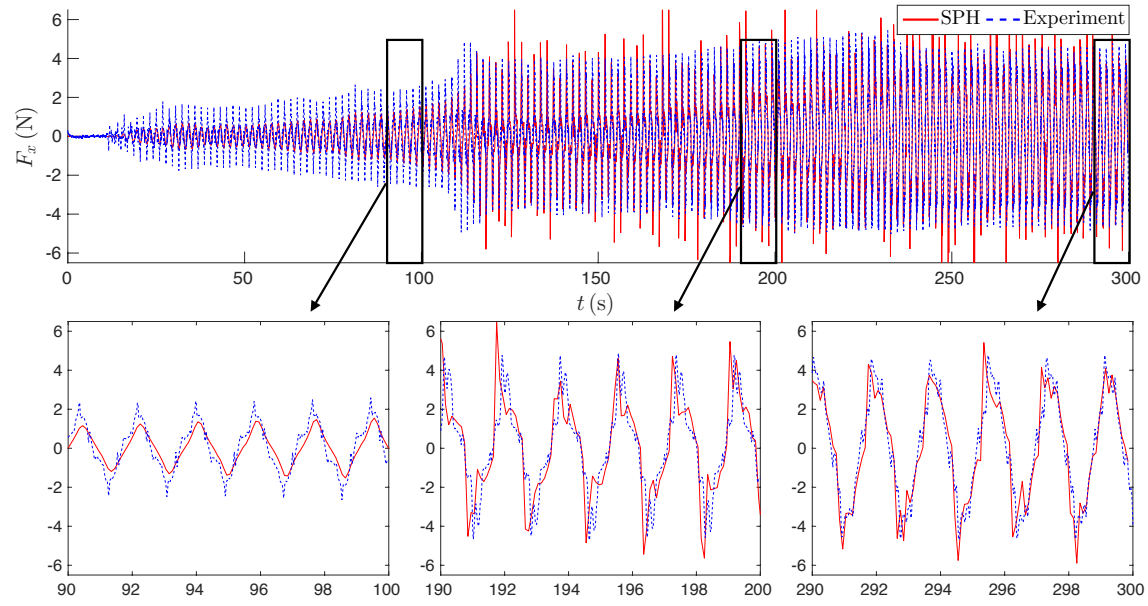


Figure 6.11: Sloshing force (N) over time (s) of the rectangular tank sloshing experiment with excitation frequency $f_{w3} = 0.55\text{Hz}$.

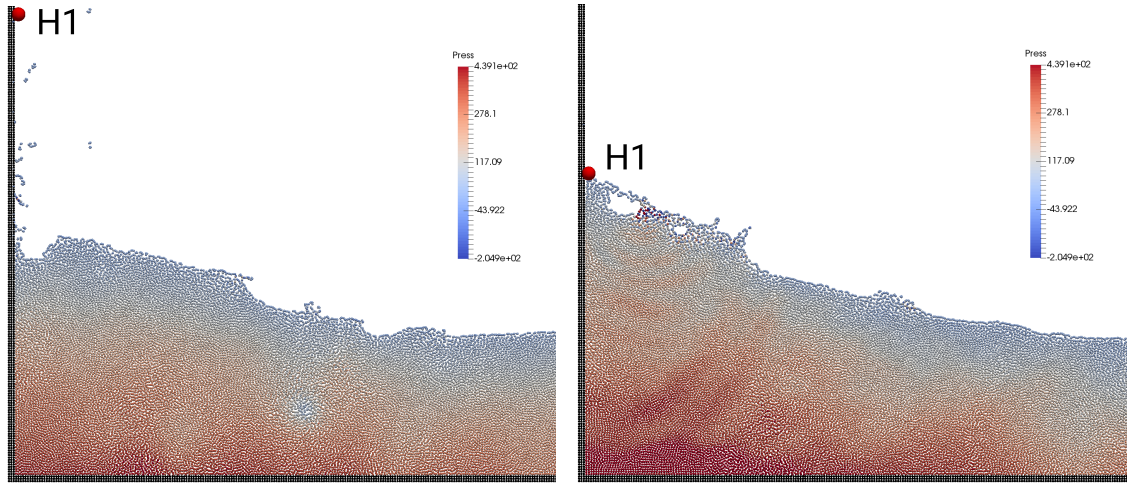


Figure 6.12: Snapshots of the SPH simulation with excitation frequency $f_{w2} = 0.46\text{Hz}$ showing height measurements at location H1 (marked by a red dot) immediately after the collapse of the wave on the wall: (left) taken at time $t = 293.1\text{s}$; (right) taken at $t = 295.3\text{s}$. Particles are coloured by their pressure values.

Snapshots representing a full sloshing cycle after reaching the steady state are provided for the cases with frequencies $f_{w2} = 0.46\text{Hz}$ and $f_{w3} = 0.55\text{Hz}$ in Figures 6.13 and 6.14, respectively. These images show the free surface from the SPH simulations at similar stages of the sloshing cycle overlaid in red. The snapshots are taken from a video recording of the experiment and the exact times corresponding to the start of the simulation at the chosen frequencies are unavailable. These snapshots serve to illustrate that the overall shape and features of the flow between the simulations and the experiment show similarities, even after the very long time they take to reach the steady state. The $f_{w1} = 0.32\text{Hz}$ case is not shown here as the dynamics are so limited that there is no added value in a visual observation of the snapshots.

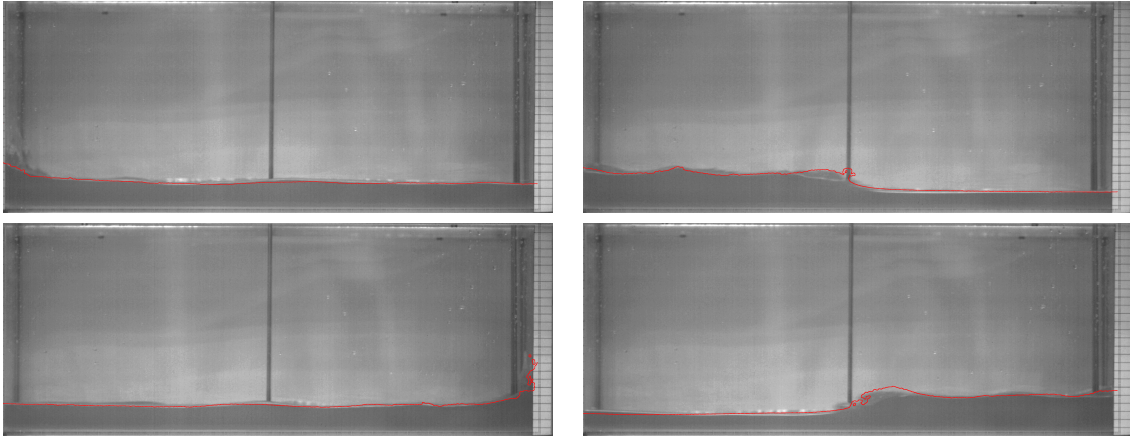


Figure 6.13: Snapshots of the rectangular tank experiment for a whole cycle after reaching a steady state under an excitation frequency of $f_{w2} = 0.46\text{Hz}$ with the SPH free-surface overlaid.

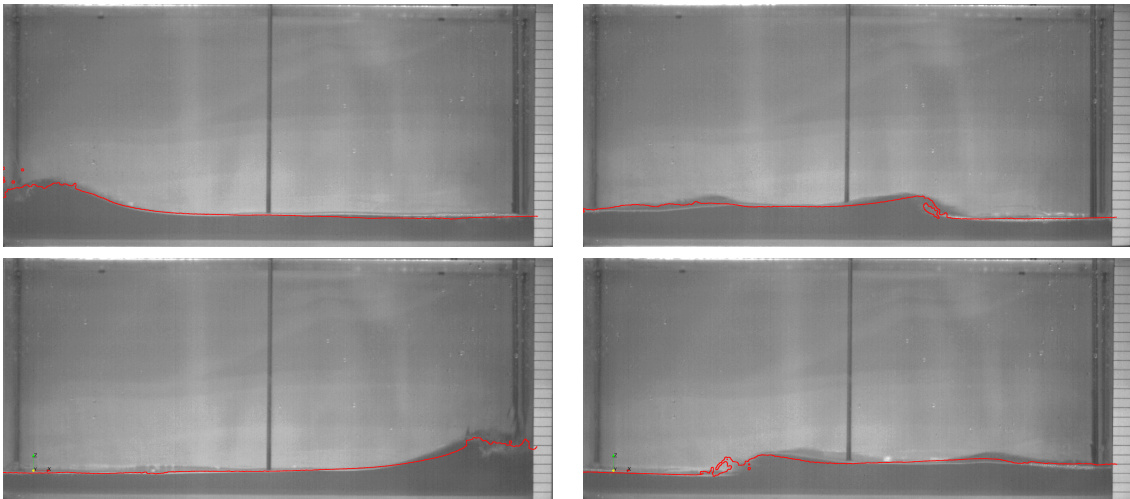


Figure 6.14: Snapshots of the rectangular tank experiment for a whole cycle after reaching a steady state under an excitation frequency of $f_{w3} = 0.55\text{Hz}$ with the SPH free-surface overlaid.

6.2 Rectangular tank three-dimensional steady-state simulations

The rectangular tank case is run in three dimensions at the natural frequency $f_w = 0.46\text{Hz}$, using the maximum displacement $A = 20\text{mm}$ from the onset and dwelling on it for a duration of 30 seconds. The simulation reached a steady state after only a few sloshing cycles, allowing for comparison against the steady-state phase of the flow in the experiment. The period of 10 – 30 seconds in the simulation is compared against the period of 280 – 300 seconds in the experiment, where both are at the steady-state phase. The simulation is run with a fairly low resolution, $\Delta x = 0.0015\text{m}$, which due to the stretching ratio resulted in 1 752 780 fluid particles and 904 821 boundary particles. This modest resolution with 20 particles in the fluid height, i.e. $h_w/\Delta x = 20$, is as low as it can reasonably be used without degrading the quality of the simulation too much to render it unusable. The fluid parameters are the same as in the two-dimensional case, but using a free-slip condition instead of the no-slip. Only the δ -SPH scheme with $\delta = 0.1$ was used and no artificial viscosity was added. The free-slip condition can be used here because the no-slip condition has negligible effect on the fluid flow at this resolution. Even though the particle spacing, and hence the smoothing length, is much larger than in the two-dimensional case, the simulation still required approximately 3 million time-steps. Considering the number of particles and the cost of using the corrected δ -SPH scheme in three dimensions, this is still extremely expensive and required approximately 5 weeks to simulate on a Nvidia Tesla K20 GPU card.

The water heights in both locations and the sloshing force show both strong agreement, with most of the secondary waves also accurately captured as observed in Figures 6.15 and 6.16. The splashes in the three-dimensional case are slightly less pronounced than in the two-dimensional case. This reinforces our conjecture to explain the cause of this discrepancy, since the fluid impacting the wall in three dimensions has more neighbours to interact with thus reducing the effect of the discretisation error.

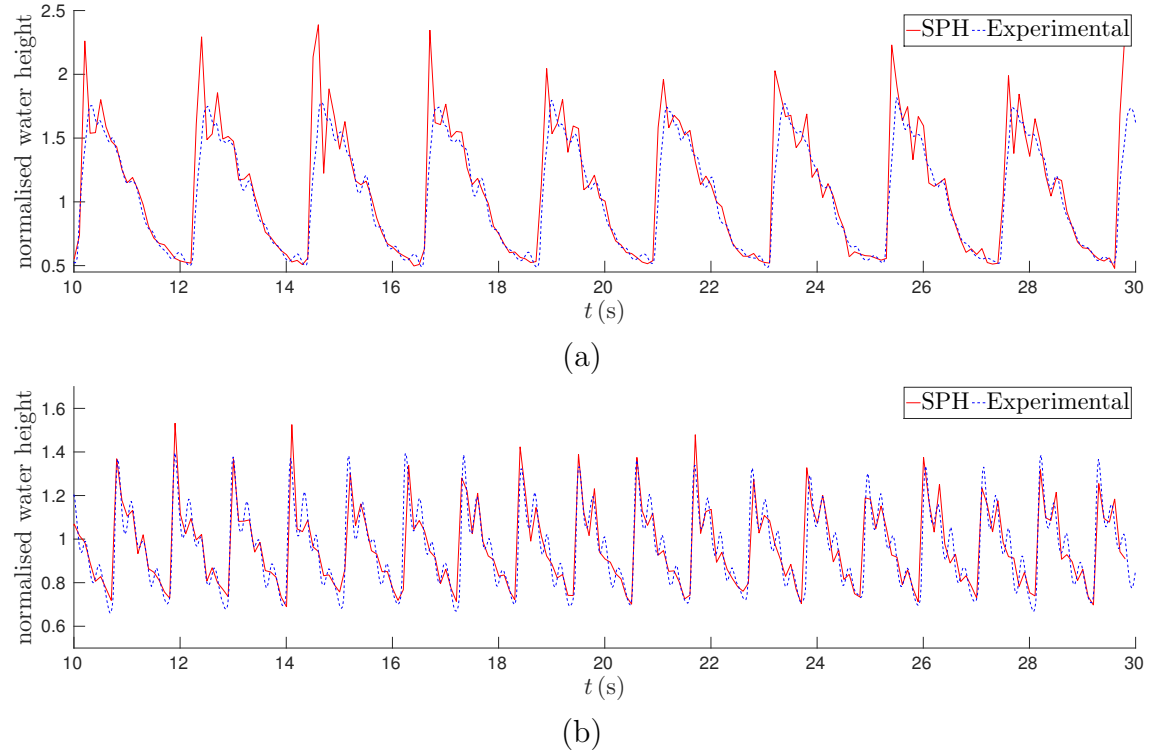


Figure 6.15: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment with an excitation frequency $f_{w2} = 0.46\text{Hz}$. Showing the time interval $280 \leq t \leq 300$ seconds in the experiment and $10 \leq t \leq 30$ seconds in the three-dimensional SPH simulation when both have reached a steady state: (a) water heights at H1; (b) water heights at H2.

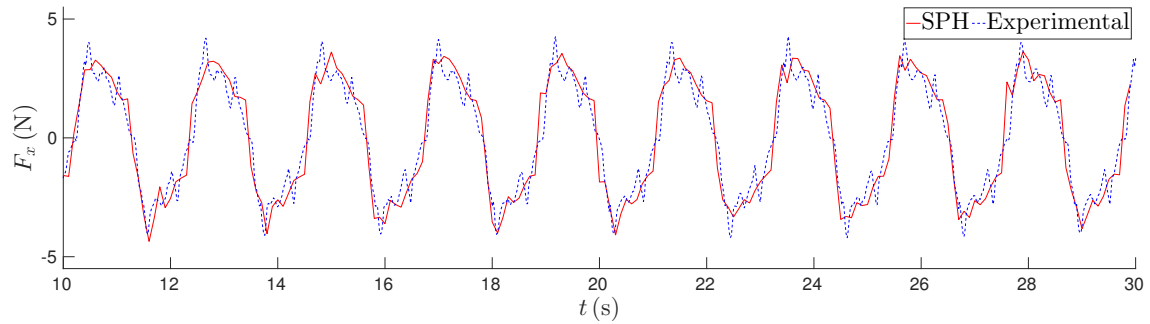


Figure 6.16: Sloshing forces (N) over time (s) in the rectangular tank sloshing experiment with an excitation frequency $f_{w2} = 0.46\text{Hz}$. Showing the time interval $280 \leq t \leq 300$ seconds in the experiment and $10 \leq t \leq 30$ seconds in the three-dimensional SPH simulation when both have reached a steady state.

To illustrate the importance of using the δ -SPH scheme in these type of simulations, an additional simulation was run without δ -SPH and using artificial viscosity (AV) with a value of $\alpha = 0.02$ for numerical stability only. The comparison of the two simulations is shown in Figures 6.17 to 6.18. The water heights at both locations are very similar and agree reasonably well with the experiment, but notable differences can be observed in the density field and the sloshing force. The simulation without δ -SPH shows a much noisier density field, as expected, and some variation in the free surface shape, but it is really the overestimation of the forces which truly demonstrates the importance of using an appropriate numerical dissipation.

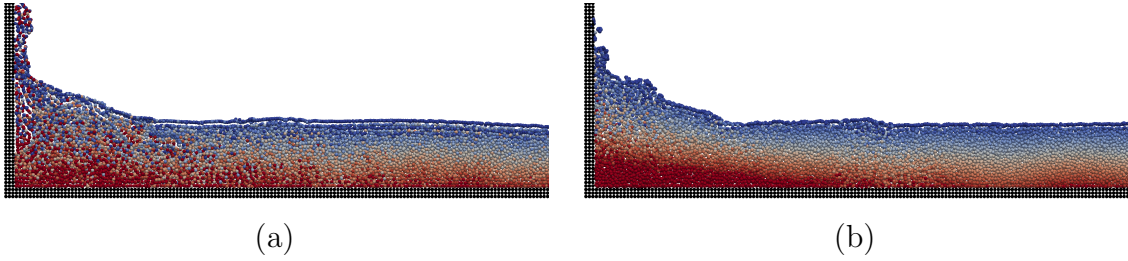


Figure 6.17: Snapshots of the density field at $t = 12.5$ s from the three-dimension SPH simulation of the rectangular tank sloshing experiment under an excitation frequency of $f_{w2} = 0.46$ Hz: (a) using artificial viscosity only with $\alpha = 0.02$; (b) using the corrected δ -SPH only with $\delta = 0.1$.

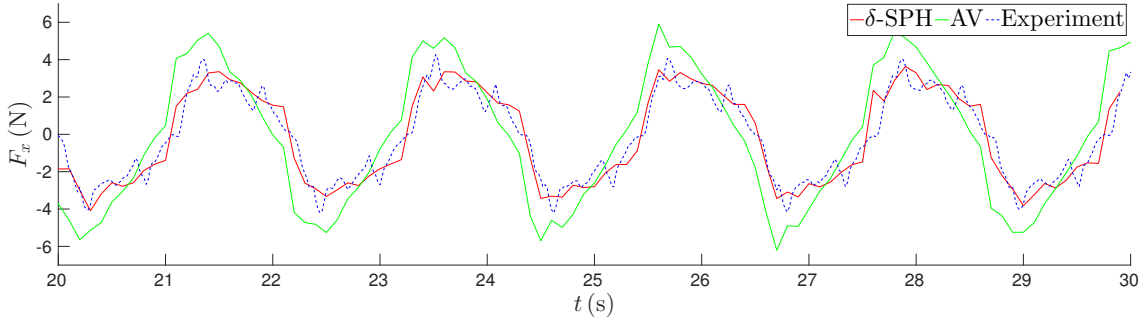


Figure 6.18: Sloshing force (N) over time (s) in the rectangular tank sloshing experiment, under an excitation frequency of $f_{w2} = 0.46$ Hz and after reaching a steady state. Comparing the three-dimensional SPH simulation using only artificial viscosity with $\alpha = 0.02$ and the SPH simulation using only the corrected δ -SPH with $\delta = 0.1$ against experimental data.

These results clearly demonstrate the ability of the SPH scheme to accurately simulate sloshing under various conditions in both two and three dimensions. They show that this version of the SPH method produces stable and accurate simulations over very long periods of time and after millions of time steps, and emphasizes the

importance of selecting the right formulation of numerical dissipation, justifying the large additional computational cost of the corrected δ -SPH scheme. Only the careful selection of the boundary condition combined with the dissipation methods, and their implementation on massively parallel architectures, such as GPUs, allowed the SPH method to accurately simulate these experiments. Using this validated SPH method is now possible to attempt the more challenging and realistic simulation of sloshing in a pill-shaped tank.

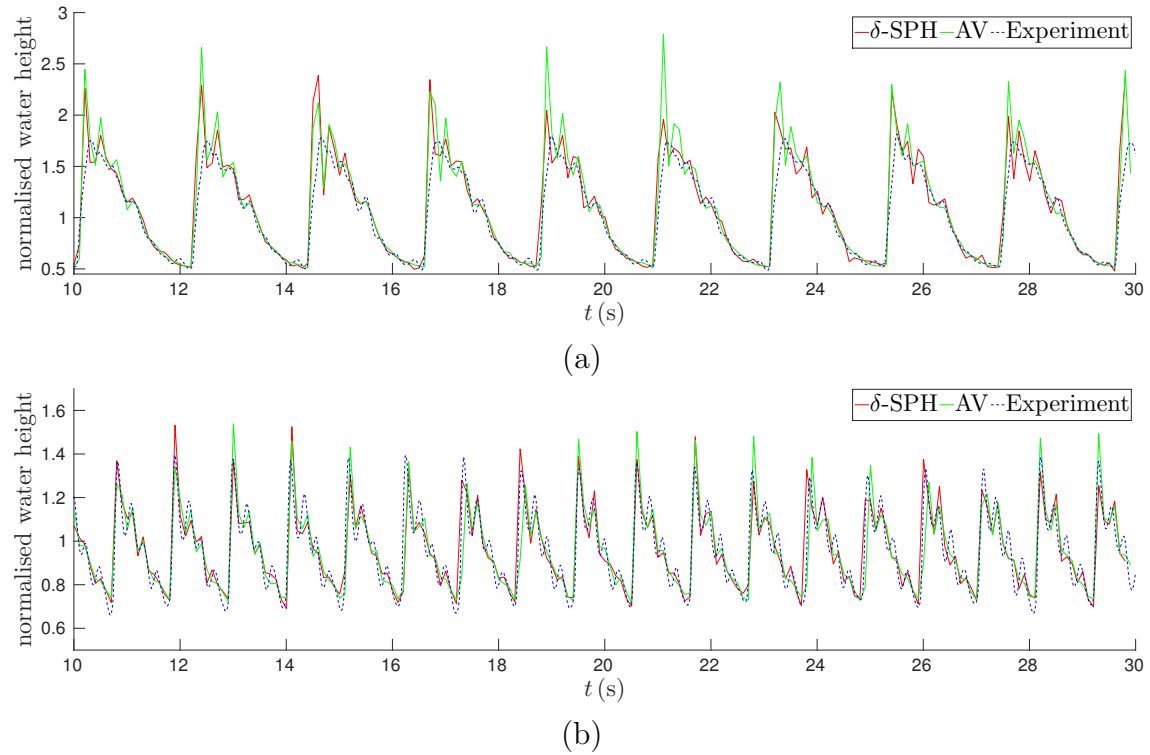


Figure 6.19: Water heights normalised by h_w over time (s) in the rectangular tank sloshing experiment, under an excitation frequency of $f_{w2} = 0.46\text{Hz}$ and after reaching a steady state. Comparing the three-dimensional SPH simulation using only artificial viscosity with $\alpha = 0.02$ and the SPH simulation using only the corrected δ -SPH with $\delta = 0.1$ against experimental data: (a) water heights at H1; (b) water heights at H2.

6.3 Sloshing in a pill-shaped tank

The pill-shaped tank experiment cycles through increasing frequencies ranging from $f_w = 1.4\text{Hz}$ to $f_w = 2.67\text{Hz}$ over a duration of 11 minutes and with a displacement $A = 20\text{mm}$. Several frequencies, starting at a frequency $f_w = 1.5\text{Hz}$, were dwelt on for a period of 60 seconds each before moving on to the next one in a continuous sweeping, as seen in Figure 6.20. Here, no steady state was reached for any of the frequencies, which made it impossible to simulate only a portion of the experiment. Hence, the simulation was run from rest and followed the experimental acceleration data during the initial build up in displacement and the first dwelling stage for the frequency $f_w = 1.5\text{Hz}$. The maximum displacement was reached after approximately 28 seconds and was kept constant afterwards as displayed in Figure 6.21. The dwelling on the first frequency started around 50 seconds. The total simulated time was 100 seconds, covering the vast majority of the dwelling period. During this period the fluid behaviour transitions from still into violent sloshing, exhibiting swirling, wave breaks, and periods of low motion in between, providing an excellent case for assessing the ability of the SPH scheme to capture all these different phenomena and validating the code.

It is probably worth noting that this experiment was only attempted once with these settings. It would be of great interest to see if any variation would occur between repeat experiments, as it is hard to believe that with the complexity of the flow a one-to-one correspondence would be obtained. It might be beneficial to conduct the experiment several times and average the forces to obtain a more conclusive comparison against the SPH simulation.

The simulation was run with a resolution of $\Delta x = 0.004\text{m}$, resulting in 459 020 fluid particles and 582 792 boundary particles. The large number of boundary particles is due to the pre-processing tool used, which enclosed the pill inside a rectangular shape. The additional boundary particles only affect the neighbour searching, where they are marked to be excluded from the particle interactions, adding only a marginal computational cost. Fluid properties are identical to the rectangular case, with $\rho_0 = 1\,000\text{kg/m}^3$, $\gamma = 7$ and $c_0 = 20\sqrt{\|\mathbf{g}h_w\|}\text{m/s}$. A free-slip boundary condition was applied and only the corrected δ -SPH scheme with $\delta = 0.1$ was used for numerical dissipation. This simulation required around 12×10^6 time-steps for the 100 seconds and took approximately 8 weeks to run on a Nvidia Tesla K40 GPU card.

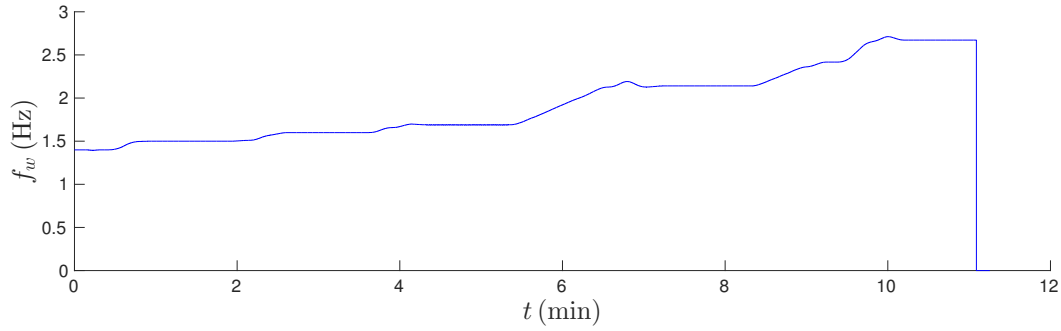


Figure 6.20: Time history of the excitation frequency (Hz) over the full duration (s) for the pill-shaped tank sloshing experiment.

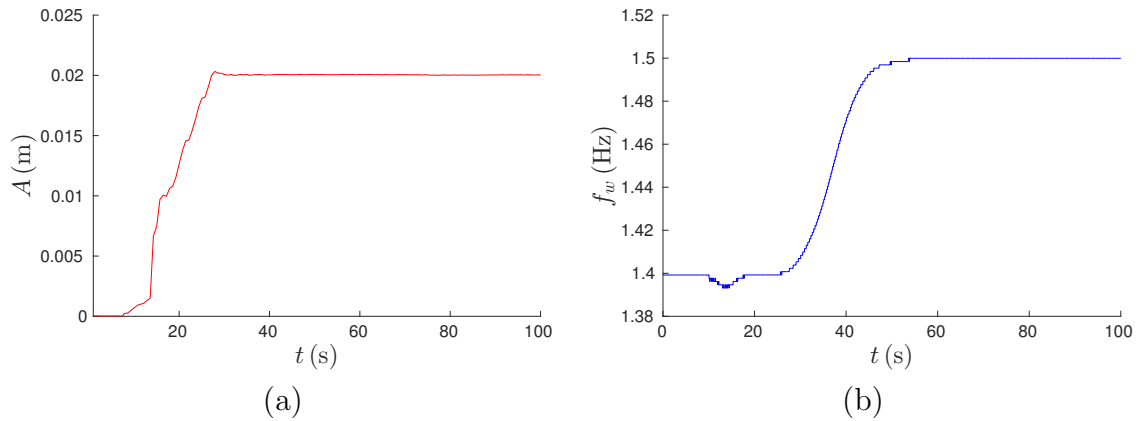


Figure 6.21: Time histories of the peak displacement (a) and excitation frequency (b) for the pill-shaped tank sloshing experiment over one hundred seconds of the SPH simulation.

Figures 6.22 (a), (b) and (c) show the x , y and z components of the force, denoted as F_x , F_y and F_z respectively, for the full duration of the simulation. Due to the complexity of the flow, and the inherent difficulty in measuring quantities such as water heights, an exact one-to-one correspondence is not expected, however many important features are still captured.

The first notable feature is the build up from low sloshing motion into violent sloshing, occurring during the increase in the displacement amplitude from around 10 to 35 seconds. During this time interval, stronger waves are gradually generated in a simple sway motion aligned with the x -axis. These can be identified in the F_x graph in Figure 6.23. From around 20 seconds, when the wave grows larger, F_z also shows good agreement. The force component F_y showed very little change as the motion was contained in the x - z plane with no swirling at this stage.

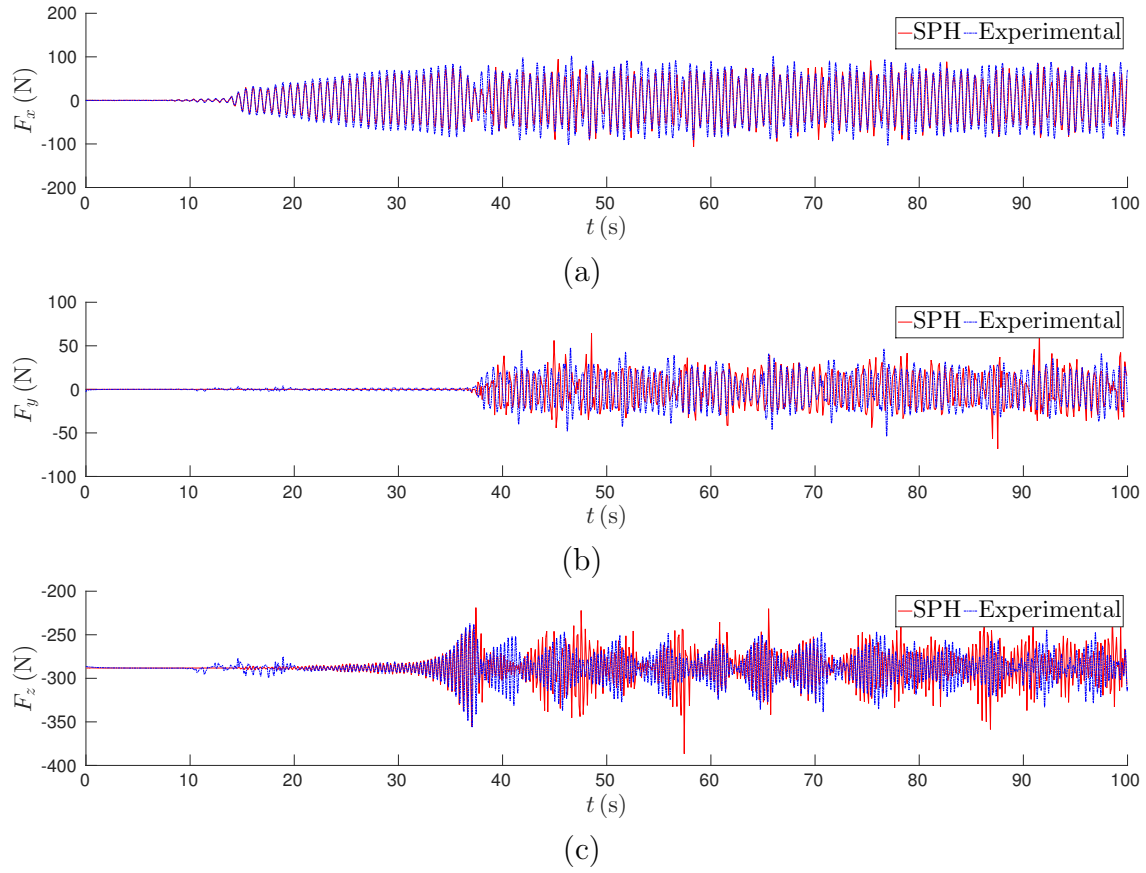


Figure 6.22: Sloshing force components (N) over time (s) for the pill-shaped tank sloshing experiment, showing the full one hundred seconds duration: (a) force in the x -axis direction F_x ; (b) force in the y -axis direction F_y ; (c) force in the z -axis direction F_z .

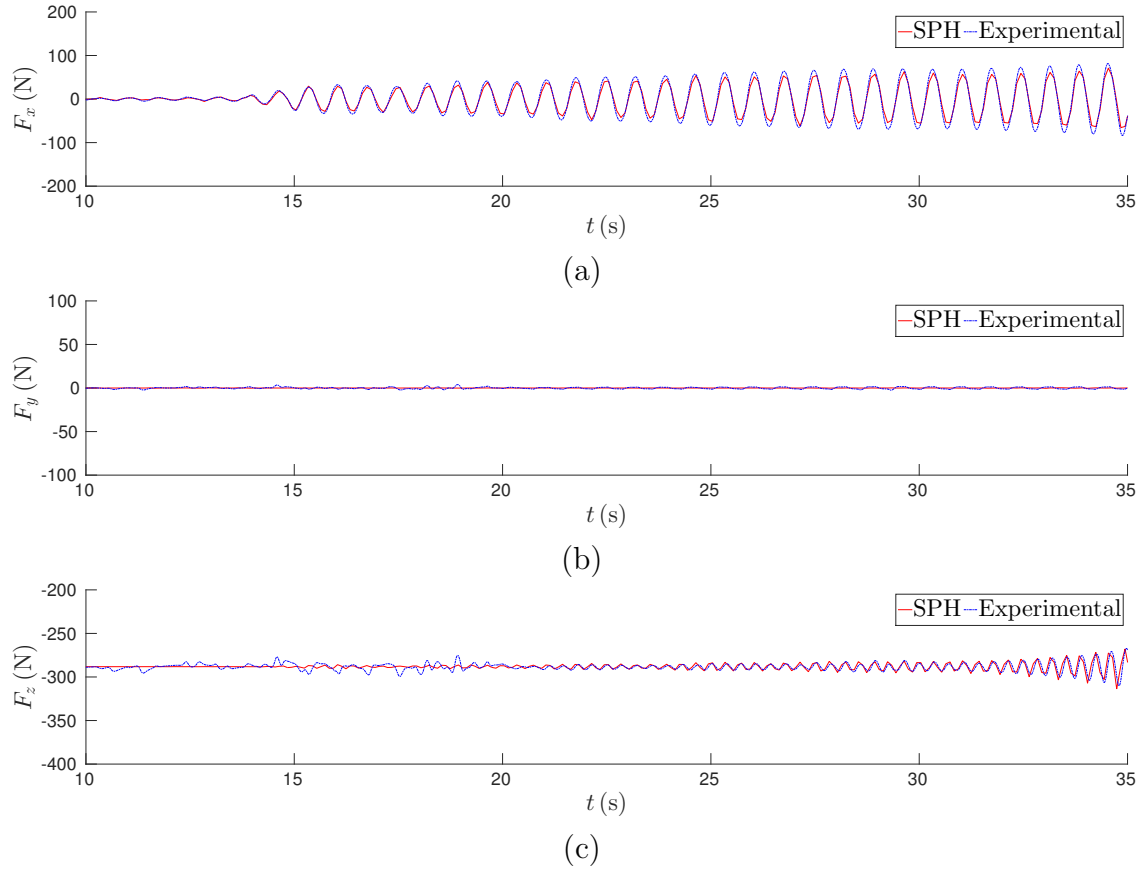


Figure 6.23: Sloshing force components (N) over time (s) for the pill-shaped tank sloshing experiment, showing the time interval $10 \leq t \leq 35$ seconds: (a) force in the x -axis direction F_x ; (b) force in the y -axis direction F_y ; (c) force in the z -axis direction F_z .

The second notable feature, and possibly the most significant, is the transition from the simple sway motion into wave breaking and the triggering of the swirling motion. This transition occurred in both the simulation and experiment at approximately 37.1 seconds. Figure 6.24 shows that, at this time, F_x and F_z exhibit a sudden decline in amplitude, while F_y suddenly grows. This sudden increase indicates the triggering of the swirling motion, as the flow is no longer contained in the x - z plane. The frequency of F_y is somewhat out of phase but with similar magnitude, while F_z has the correct frequency but shows a different pattern in the collapse of the amplitude after the transition. The component F_x is still dominated by the motion of the tank and shows a strong resemblance between the simulation and experiment apart from some slight deviation in the amplitude. After the transition, both F_y and F_z go in and out of phase rapidly, sometimes showing opposite behaviour which indicates the presence of waves or swirling in the opposite direction. To better visualise the

transition, snapshots from the experiment and simulation from 36.8 to 37.6 seconds are shown side by side in Figure 6.25. The wave breakage which triggers the swirling motion is clearly observed. These images show a strong resemblance in fluid shape, and reinforce the interpretation of the flow phenomena obtained from the force data.

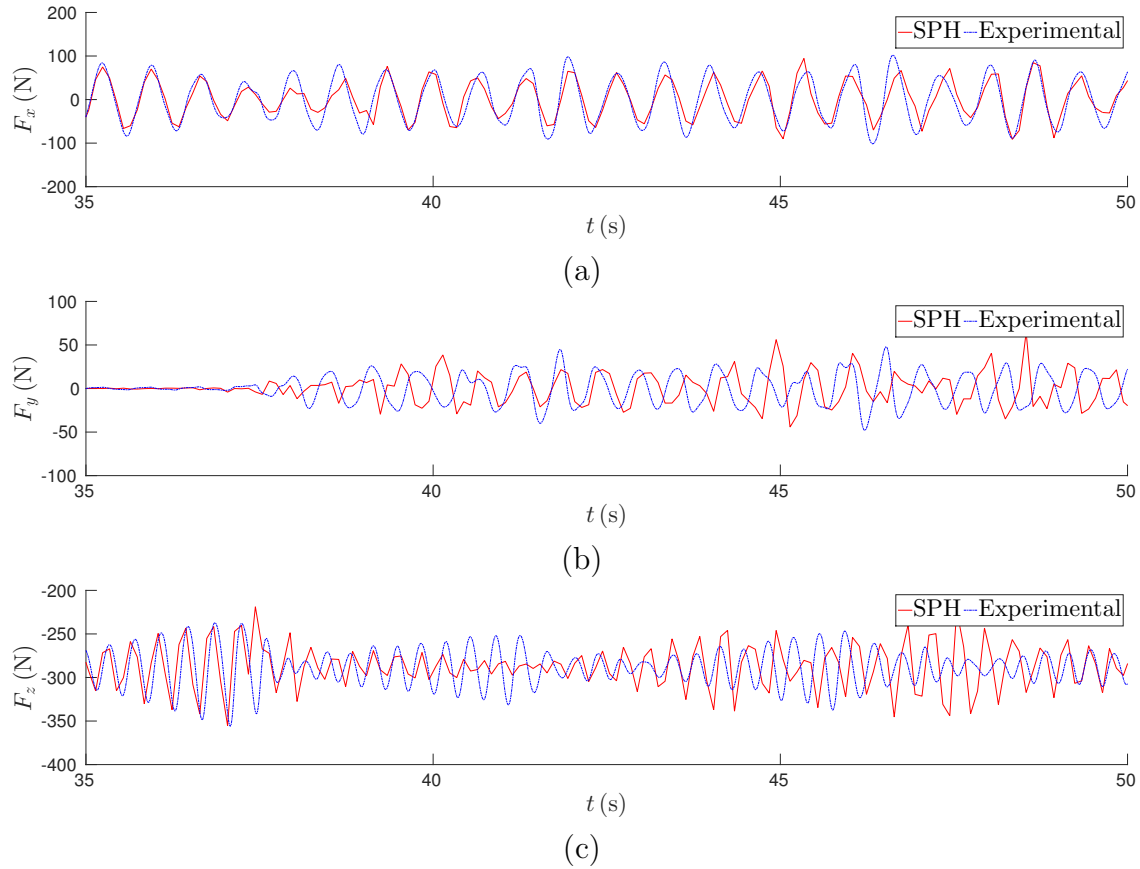


Figure 6.24: Sloshing force components (N) over time (s) for the pill-shaped tank sloshing experiment, showing the time interval $35 \leq t \leq 50$ seconds: (a) force in the x -axis direction F_x ; (b) force in the y -axis direction F_y ; (c) force in the z -axis direction F_z .

Further notable phenomena occur between 58 to 71 seconds. During this time period, the fluid undergoes rapid changes in the direction of rotation, causing wave heights to dampen significantly at approximately 64 seconds and again around 72 seconds. At this time, the component F_y is in phase with the experiment for the longest time period. In fact, this is the only period when all the three force components match the experiment simultaneously, as shown in Figure 6.26.

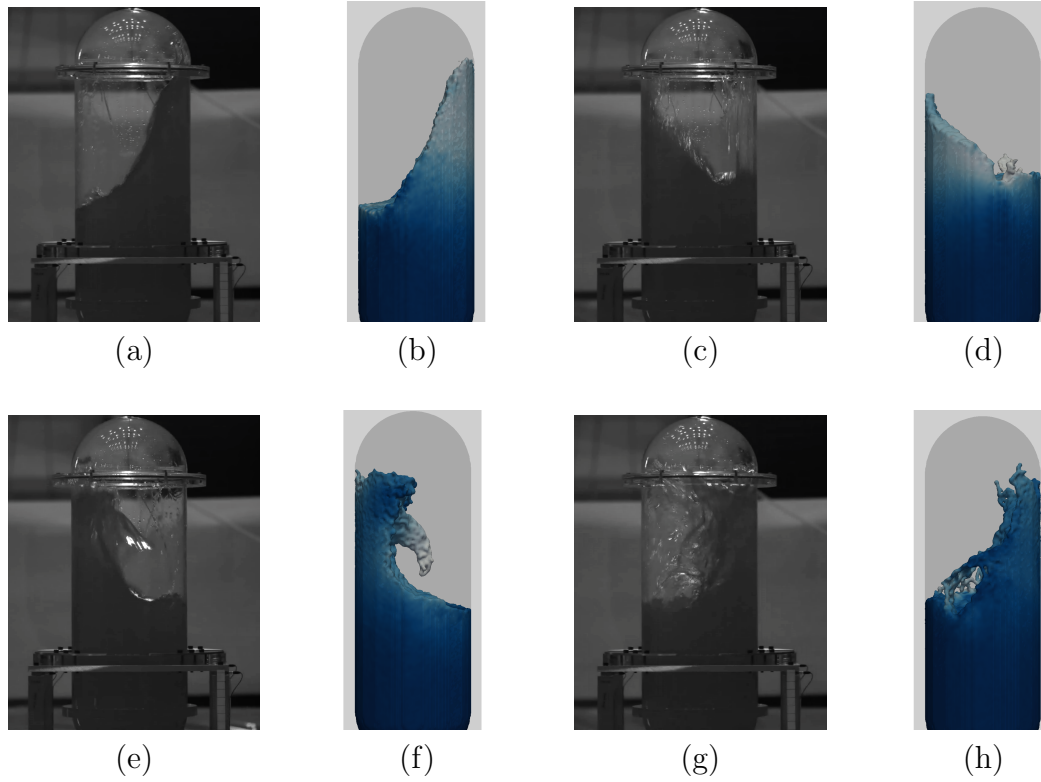


Figure 6.25: Side-by-side snapshots from the pill-shaped tank sloshing experiment alongside the SPH simulation. Taken around the time the breaking waves trigger the transition into swirling: (a) – (b) $t = 36.8$ seconds; (c) – (d) $t = 37.1$ seconds; (e) – (f) $t = 37.3$ seconds; (g) – (h) $t = 37.6$ seconds.

The last 20 seconds of the simulation are shown in Figure 6.27. This period of time shows a typical behaviour of the flow and the best agreement between the simulation and the experiment. The force component F_x stays similar as in all cases, while F_y and F_z go in and out of phase as the simulation and the experiment take different rotational directions. The force component F_y shows overall similar amplitude, with the frequencies switching quickly from being synchronized to be in opposite phase. This is assumed to be due to the swirling going in opposite directions. On the other hand, the component F_z exhibits many peaks and troughs, which are assumed to be related to instances of wave build-up and break-down respectively. While the overall amplitudes of F_z are similar, the experiment shows slightly more regular sequences of peaks and troughs than the simulation.

Further analysis of the fluid motion via a discrete fast Fourier transform (FFT) and a power spectral density (PSD) permits a comparison of the frequency content of the simulations against that of the experimental measurements. The results of this frequency analysis are displayed in Figures 6.28 to 6.30 for the force components F_x ,

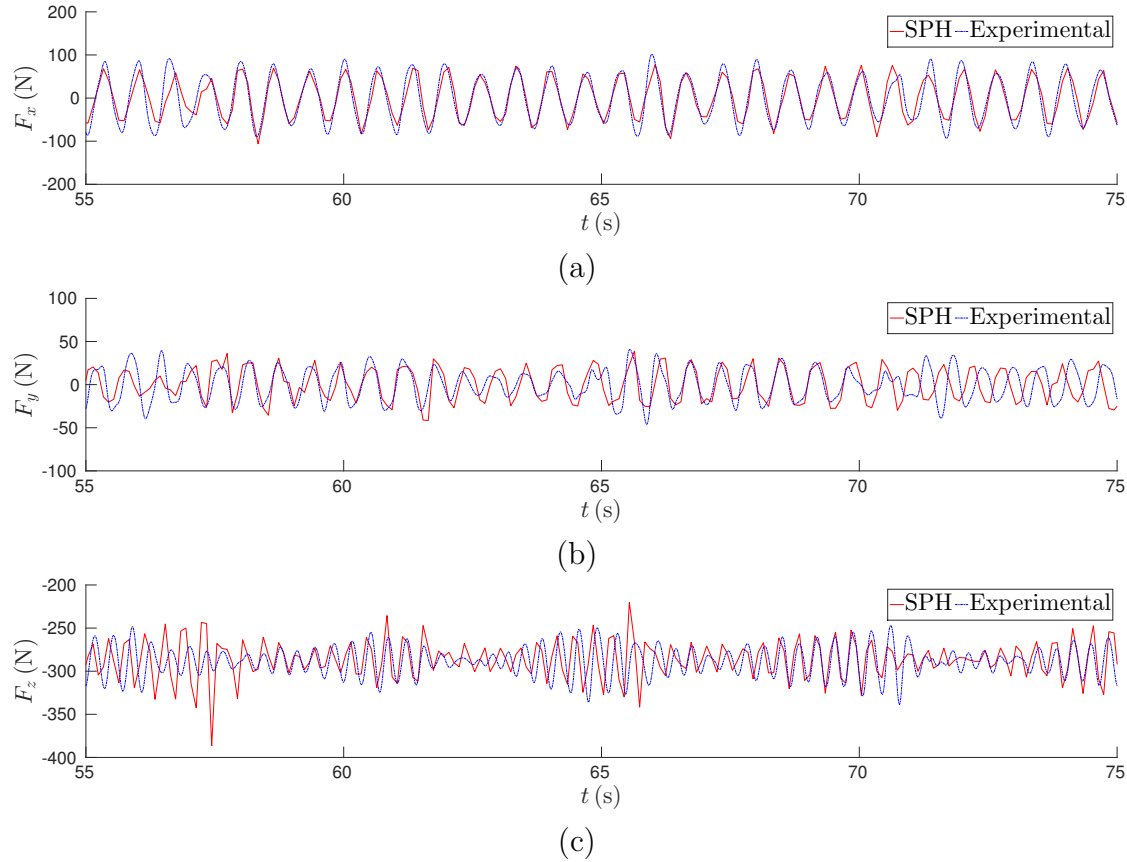


Figure 6.26: Sloshing force components (N) over time (s) for the pill-shaped tank sloshing experiment, showing the time interval $55 \leq t \leq 75$ seconds: (a) force in the x -axis direction F_x ; (b) force in the y -axis direction F_y ; (c) force in the z -axis direction F_z .

F_y and F_z respectively. The frequency spectrum of the component F_x calculated by FFT shows an almost perfect agreement of peak frequencies, which is to be expected from forcing excitation along the x -axis. A more interesting finding is that the components F_y and F_z show very similar frequency content despite the significantly different time histories. The peak frequencies in the FFT spectra are very close, with only slightly lower dominant frequencies in the simulation than in the experiment, but are almost identical in the PSD spectra. The simulation also shows slightly more contribution from the less dominant frequencies than those in the experiment for all three force components. This can be inferred from the higher powers away from the peak frequencies in the PSD. As the components F_y and F_z are not driven by the external force, the strong agreement between simulation and experiment obtained in the frequency domain analysis provides further evidence of the ability of the proposed SPH method to accurately capture the important features of such complex flows.

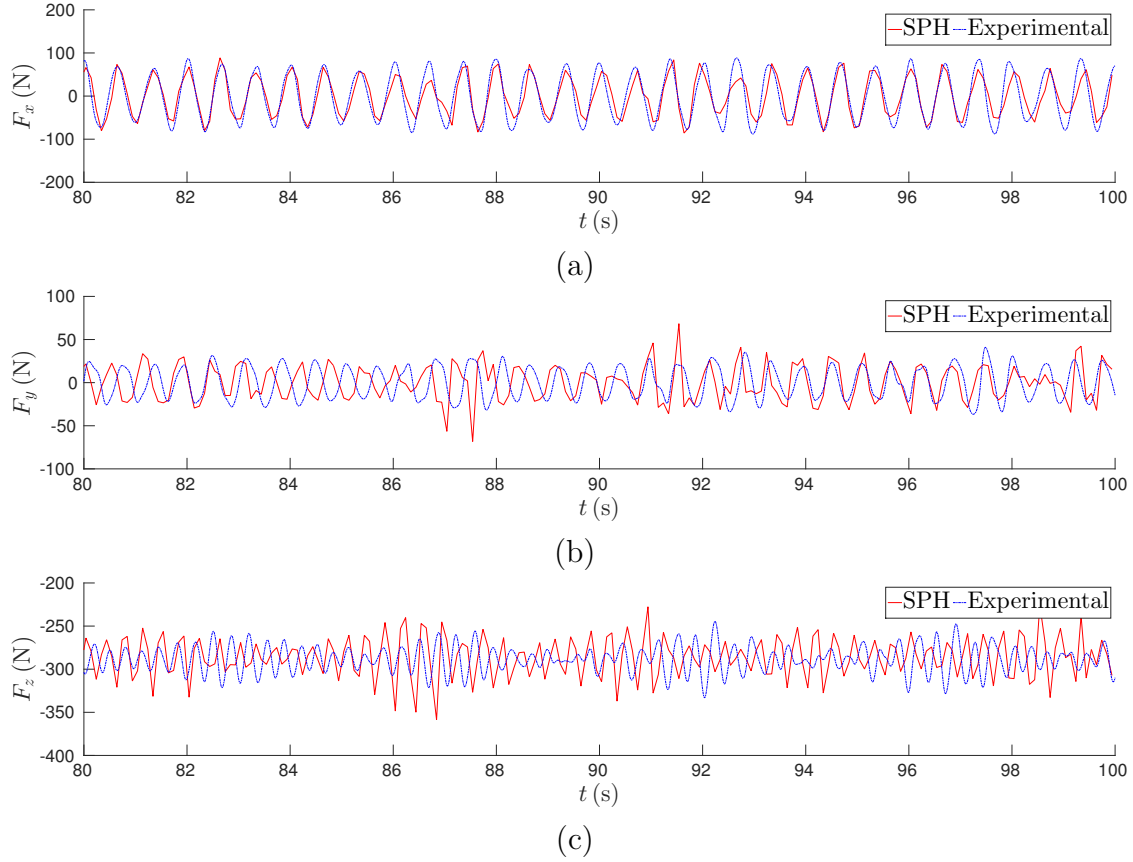


Figure 6.27: Sloshing force components (N) over time (s) for the pill-shaped tank sloshing experiment, showing the time interval $80 \leq t \leq 100$ seconds: (a) force in the x -axis direction F_x ; (b) force in the y -axis direction F_y ; (c) force in the z -axis direction F_z .

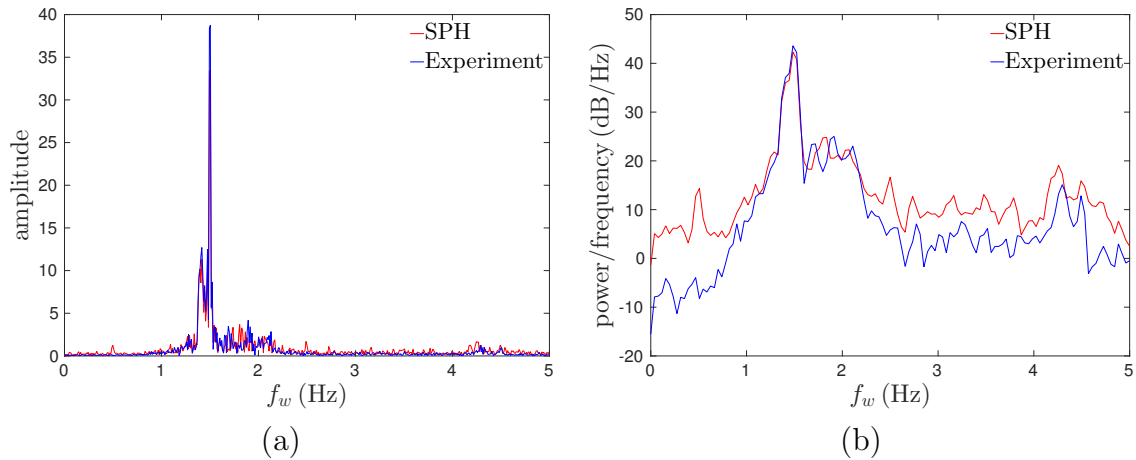


Figure 6.28: The frequency domain of the x -axis sloshing force component for the pill-shaped tank sloshing experiment. Comparison of the SPH simulation against the experiment: (a) using fast Fourier transform (FFT); (b) using the power spectral density (PSD).

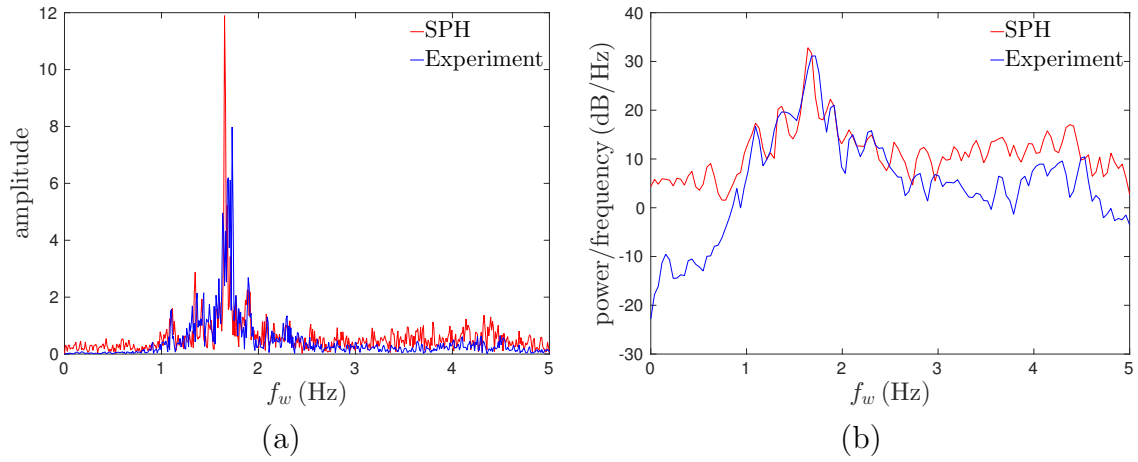


Figure 6.29: The frequency domain of the y -axis sloshing force component for the pill-shaped tank sloshing experiment. Comparison of the SPH simulation against the experiment: (a) using fast Fourier transform (FFT); (b) using the power spectral density (PSD).

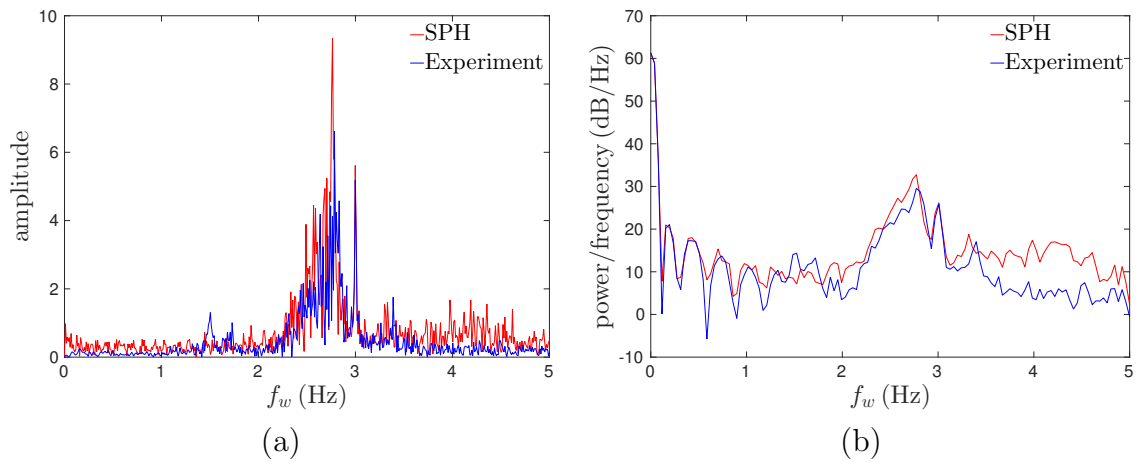


Figure 6.30: The frequency domain of the z -axis sloshing force component for the pill-shaped tank sloshing experiment. Comparison of the SPH simulation against the experiment: (a) using fast Fourier transform (FFT); (b) using the power spectral density (PSD).

Chapter 7

Conclusions

This thesis has provided an investigation of the capabilities of the state-of-the-art in SPH to simulate liquid sloshing and shown that, with careful consideration, SPH is capable of accurately simulating long duration transient sloshing in various scenarios.

The investigation identified three areas of critical importance for the method to give stable and accurate results for the sloshing problems. These are the treatment of solid wall boundary conditions, numerical dissipation and the efficient parallel implementation such as on a GPU. Each of these issues has been dealt with over the years in the SPH literature, but rarely have they been put together in an engineering context. The most suitable solutions to these issues were identified here and described in detail.

The fixed ghost particle method of Adami et al. 2012 [3] was selected for the implementation of wall boundary conditions and described in Section 3.4.2. This method includes a force balance correction which greatly improves the smoothness of the fluid properties adjacent to the solid walls. It is also simple to implement and numerically efficient.

The corrected δ -SPH scheme proposed in Antuono et al. 2010 [8] is identified as being the most suitable form of numerical dissipation for long duration simulations and is described in Section 3.5.3. This scheme adds a dissipative term in the mass conservation equation which stabilises the SPH method and helps reduce the commonly found spatial oscillations of pressure. Moreover it takes particular care to conserve volume and it reduces the dissipation of energy unlike alternative approaches that are commonly used.

To address the efficient parallelisation requirements, the fixed ghost boundary condition and the corrected δ -SPH scheme were implemented in a GPU code named

DualSPHysicsGPM which is a modified version of the open-source SPH solver DualSPHysics (version 4) [28]. Details of the code are found in Section 4.4. DualSPHysicsGPM, using the fixed ghost particles boundary condition, was shown to be as efficient as DualSPHysics. However, the corrected δ -SPH scheme requires a very large increase in computational cost, between five to seven times depending on the specific problem and simulation parameters.

The combination of the selected schemes presented in the thesis and their implementation in DualSPHysicsGPM was thoroughly validated and verified through a series of tests described in Chapter 5. Comparisons were made against analytic solutions of a hydrostatic tank and linear sloshing, and against experimental data of a simple two-dimensional dam break and three-dimensional dam break with a box in the tank. The test cases focused on the fixed ghost boundary condition, the corrected δ -SPH scheme and the overall combination, comparing them against published results or the DualSPHysics code where possible. These comparisons show that DualSPHysicsGPM can not only resolve the various test cases in a satisfactory manner, but in many instances provides significant improvements over DualSPHysics.

In particular, the fixed ghost particle method shows significant improvements on the pressure noise, mostly in regions close to the solid walls, corners and the triple point (solid wall, fluid and free surface), and a slightly more ordered particle distribution on the free surface. The method also eliminates a non-physical gap between the wall and the fluid which appears in several other boundary conditions. Lastly, it shows an improvement of an order of magnitude in total energy conservation during the first few seconds of a simulation.

The corrected δ -SPH scheme is the best option to reduce spatial pressure oscillation for long durations. Additionally the corrected δ -SPH showed, in the hydrostatic validation test, advantages over the use of artificial viscosity over prolonged durations, specifically by reducing the oscillations in potential energy. These are damped much faster and are restricted to only a few particles on the free surface, rather than effecting the bulk of the fluid. The use of the corrected δ -SPH scheme was also required to obtain correct sloshing forces, despite having negligible effects on the wave heights: this was an unexpected find.

The main goal of developing DualSPHysicsGPM was to accurately simulate sloshing in real engineering problems and reproduce the experiment by ESA/ESTEC described in Chapter 6. This experiment required the ability to simulate low fill ratio liquid sloshing in a rectangular tank and sloshing with swirling wave motion in a

pill-shaped tank. The rectangular tank test, which was run for a duration of three hundred seconds each time, provided important insights of the SPH method. It was the catalyst that lead to the investigation of the three areas of concern described above. It also highlighted the need for double precision to represent the positions of the particles. It should serve as a warning to carefully check whether seemingly simple cases might suffer from numerical inaccuracies caused by single precision commonly favoured in GPU implementations.

DualSPHysicsGPM is capable of accurately simulating the experimental sloshing cases. The rectangular tank was simulated in two dimensions for frequencies below, at and above the first natural sloshing mode over long durations. Both the wave heights and the sloshing forces have been compared during transient and steady-state phases of the experiment and showed good agreement with the experimental measurements. Further, a three-dimensional simulation at the natural frequency was validated in the steady-state regime. This simulation required just under three million particles and was run for thirty “physical” seconds - a feat that would be computationally intractable without an efficient parallel code. The three-dimensional simulation also emphasised the need for the corrected δ -SPH scheme in order to obtain an accurate sloshing force.

The pill-shaped tank simulation was validated against the experiment over a period of one hundred seconds starting from rest. Sloshing force components in the x , y , and z axis directions, as well as video footage, were used for the comparison. The SPH simulation was capable of capturing important features of the complex flow observed in the experiment. These include the build up of swaying waves, the transition into swirling waves and periods of wave build up and break down. A one to one correspondence was not anticipated for such a complex case with transient swirling motions, however comparison of the frequency domain showed very strong resemblance. As far as the author is aware, this is the first successful attempt to accurately reconstruct the whole time history for such a complex sloshing case.

7.1 Future work

The implementation of the corrected δ -SPH scheme in DualSPHysicsGPM, described in Section 4.4.2, is currently not optimised for efficiency on GPUs. The implementation requires storing a three by three matrix of double precision floats and all operations associated with the linear solver for obtaining the corrected density gra-

dients are also carried out in double precision. Performance improvements can be achieved by a better use of the GPU memory and by minimising the operations using double precision. This additional work could drastically improve the efficiency of the scheme at moderate development cost. In addition, it would be advisable to investigate alternative fourth-order numerical dissipative schemes. For example, the scheme of Fatehi and Manzari [44] is a fourth-order scheme proposed to correct the spatial pressure oscillations, but has not been validated for the type of problems of interest here.

The free-slip boundary conditions for the fixed ghost particle method, while satisfactory in practice, are not entirely correct. In the current formulation, the velocity of the boundary particles is set to zero and the viscous interactions with fluid particles neglected. The correct free-slip condition should have the velocity of a boundary particle equal in the tangent direction and opposite in the normal direction. This would require to know the direction of the normals for each boundary particle. This information is not currently available in DualSPHysics or DualSPHysicsGPM. However, while theoretically correct, in practice this method might have adverse effects. The force balance calculation of the boundary particles in equation (3.57) uses a zeroth-order re-normalisation in the denominator to account for the truncated kernel in the fluid domain. It would be advisable to investigate if using a first-order kernel correction, such as the moving least-squares [13], would improve the interpolation.

Other topics that deserves further investigation are the no-slip boundary condition and the viscosity models. A detailed comparison of the methods suggested in Section 3.5.2, and possibly other schemes, such as the double summation approach suggested in reference [146] should be undertaken in order to understand their effects on long duration sloshing cases.

The last topic relating to the boundary condition concerns the issue of isolated fluid particle penetrating the walls. This error is described in detail in Section 3.4.3, which puts forward a hypothesis of the source for this error and a suggestion on how to address it. A further investigation into these issues is highly desirable and it would also contribute to enhancing other types of boundary conditions too.

The last suggestion for further development is support for multi-GPU implementations. DualSPHysics, and subsequently DualSPHysicsGPM, are currently limited to the use of a single GPU card. As it stands, the run times for long duration three-dimensional cases are possibly too long for every day engineering practice. Initial work in this regard has been done [37] and it is anticipated to be included in future

releases of DualSPHysics.

To make DualSPHysicsGPM an adequate tool for cryogenic micro-gravity sloshing, additional physical models must be included. These are, amongst others, multi-fluid for liquids and gases, surface tension, thermal boundaries, heat transfer and phase change.

Liquid and gas multi-fluid formulations with surface tension have been studied in SPH for a range of physical problems. However, difficulties can arise in the basic SPH formulation, particularly for high density ratios. A promising formulation for multi-fluid simulations is given by Hu and Adams [66] that is claimed by the authors to deal with high density ratios. A surface tension term was later included in that method by Adami et al. [2]. These two formulations were included in a beta version of DualSPHysics version 3 by Mokos et al. [95] and would provide a good starting point for the investigation of multi-fluid and surface tension capabilities for complex sloshing problems. However, the surface tension formulation is quite simple, using a colour function to obtain the gradient of the free surface and assumes a constant contact angle, which might not be sufficient for cryogenic fuel sloshing in micro-gravity. Further, Mokos et al. report several limitations and difficulties with the method, such as the formations of large gaps, which require novel particle shifting techniques to address. These methods have not been validated for surface tension dominated flows or under micro-gravity assumptions, and their applicability to these conditions is not guaranteed.

Heat transfer will require the addition of the energy conservation equation into the governing equations and thermal boundaries. These topics have rarely been dealt with in the SPH methodology. Some examples of this type of work can be found in references [21, 71, 79, 140]. However, the scope of problems in these works are limited to simple cases with a single fluid and no complex fluid behaviour. Heat transfer with multi-fluids provides a further complexity due to different constitutive equations for each phase. This makes modelling phase-changes extremely challenging, particularly for cryogenic fuels where the constitutive equations for the gas phase are not well defined. Indeed, as far as the author is aware, phase change from liquid to gas has not been modelled with SPH at the time of writing. Further, coupling thermal conductivity with surface tension and possibly incorporating its effect on the contact angle is required. That too has not been sufficiently explored with the SPH methodology.

Bibliography

- [1] H.N. Abramson. The dynamic behavior of liquids in moving containers with applications to space vehicle technology. Technical report, NASA SP-106, 1966.
- [2] S. Adami, X.Y. Hu, and N.A. Adams. A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation. *Journal of Computational Physics*, 229(13):5011–5021, 2010.
- [3] S. Adami, X.Y. Hu, and N.A. Adams. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*, 231(21):7057–7075, 2012.
- [4] T. Amada, M. Imura, Y. Yasumotoa, Y. Yamabe, and K. Chihara. Particle-based fluid simulation on GPU. In *ACM Workshop on General-Purpose Computing on Graphics Processors*, Los Angeles, California, USA, 2004. ACM.
- [5] A. Amicarelli and G. Agate. A 3D fully Lagrangian smoothed particle hydrodynamics model with both volume and surface discrete elements. *International Journal for Numerical Methods in Engineering*, 95(5):419–450, 2013.
- [6] ANSYS. CFX and Fluent. www.ansys.com. [Online; accessed 17-October-2016].
- [7] M. Antuono, A. Colagrossi, and S. Marrone. Numerical diffusive terms in weakly-compressible SPH schemes. *Computer Physics Communications*, 183(12):2570–2580, 2012.
- [8] M. Antuono, A. Colagrossi, S. Marrone, and D. Molteni. Free-surface flows solved by means of SPH schemes with numerical diffusive terms. *Computer Physics Communications*, 181(3):532–549, 2010.

- [9] Near Earth Asteroid Rendezvous Anomaly Review Board. The NEAR Rendezvous Burn Anomaly of December 1998. Technical report, The Johns Hopkins University Applied Physics Laboratory, 1999.
- [10] OpenMP Architecture Review Board. OpenMP. openmp.org/. [Online; accessed 17-October-2016].
- [11] J. Bonet and S. Kulasegaram. A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Applied Mathematics and Computation*, 126(2-3):133–155, 2002.
- [12] J. Bonet, S. Kulasegaram, M.X. Rodriguez-Paz, and M. Profit. Variational formulation for the smooth particle hydrodynamics (SPH) simulation of fluid and solid problems. *Computer Methods in Applied Mechanics and Engineering*, 193(12–14):1245–1256, 2004.
- [13] J. Bonet and T.-S.L. Lok. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering*, 181(1):97–116, 1999.
- [14] J. Bonet and M.X. Rodriguez-Paz. Hamiltonian formulation of the variable- h SPH equations. *Journal of Computational Physics*, 209(2):541–558, November 2005.
- [15] H. Braess and P. Wriggers. Arbitrary Lagrangian Eulerian finite element analysis of free surface flow. *Computer Methods in Applied Mechanics and Engineering*, 190(1):95–109, 2000.
- [16] B. Buchner. *Green water on ship-type offshore structures*. PhD thesis, Delft University of Technology, 2002.
- [17] R. Canelas, A.J.C. Crespo, J.M. Domínguez, and R.M.L. Ferreira. A generalized SPH-DEM discretization for the modelling of complex multiphasic free surface flows. In *Proceedings of the 8th International SPHERIC Workshop*, 2013.
- [18] CD-Adapco. StarCCM+. mdx.plm.automation.siemens.com/star-ccm-plus. [Online; accessed 17-October-2016].
- [19] CEHINAV. AQUAghusph. canal.etsin.upm.es/aquagphusph/. [Online; accessed 17-October-2016].

- [20] J.L. Cercós-Pita. AQUAgpusph, a new free 3D SPH solver accelerated with OpenCL. *Computer Physics Communications*, 192:295–312, 2015.
- [21] P.W. Cleary. Modelling confined multi-material heat and mass flows using SPH. *Applied Mathematical Modelling*, 22(12):981–993, 1998.
- [22] A. Colagrossi, M. Antuono, and D. Le Touzé. Theoretical considerations on the free surface role in the smoothed-particle-hydrodynamics model. *Physical Review E*, 79(5):056701, 2009.
- [23] A. Colagrossi and M. Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191(2):448–475, 2003.
- [24] A. Colagrossi, C. Lugni, M. Greco, and O.M. Faltinsen. Experimental and numerical investigation of 2D sloshing with slamming. In *19th International Workshop on Water Waves and Floating Bodies*, Cortona, Italy, 2004.
- [25] Sgi Corp. OpenFoam. www.openfoam.com. [Online; accessed 17-October-2016].
- [26] Cray, CAPS, Nvidia, and PGI. OpenACC. www.openacc.org/. [Online; accessed 17-October-2016].
- [27] A.J.C. Crespo, J.M. Domínguez, A. Barreiro, M. Gómez-Gesteira, and B.D. Rogers. GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods. *PLoS one*, 6(6):e20685, 2011.
- [28] A.J.C. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal. DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications*, 187:204–216, 2015. DualSPHysics v4 is available at: www.dual.sphysics.org.
- [29] A.J.C. Crespo, M. Gómez-Gesteira, and R.A. Dalrymple. Boundary conditions generated by dynamic particles in SPH methods. *Computers, Materials, & Continua*, 5(3):173–184, 2007.
- [30] A.J.C. Crespo, J.C. Marongiu, E. Parkinson, M. Gómez-Gesteira, and J.M. Domínguez. High Performance of SPH Codes: Best approaches for efficient

- parallelization on GPU computing. In *Proceedings of the 4th International SPHERIC Workshop*, pages 69–76, 2009.
- [31] S.J. Cummins and M. Rudman. An SPH projection method. *Journal of computational physics*, 152(2):584–607, 1999.
- [32] R. Davé, J. Dubinski, and L. Hernquist. Parallel TreeSPH. *New Astronomy*, 2(3):277–297, 1997.
- [33] W. Dehnen and H. Aly. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society*, 425(2):1068–1082, 2012.
- [34] L. Delorme, A. Colagrossi, A. Souto-Iglesias, R. Zamora-Rodríguez, and E. Botia-Vera. A set of canonical problems in sloshing. Part I: Pressure field in forced roll-comparison between experimental results and SPH. *Ocean Engineering*, 36(2):168–178, 2009.
- [35] L. Delorme and A. Souto-Iglesias. Sloshing loads simulation in LNG tankers with SPH. In *International Conference on Computational Methods in Marine Engineering*, 2005.
- [36] R. Dolbeau, S. Bihan, and F. Bodin. HMPP: A hybrid multi-core parallel programming environment. In *Workshop on general purpose processing on graphics processing units (GPGPU 2007)*, volume 28, 2007.
- [37] J.M. Domínguez. *DualSPHysics: Towards High Performance Computing using SPH technique*. PhD thesis, Environmental Physics Laboratory, University of Vigo, September 2014.
- [38] J.M. Domínguez, A.J.C. Crespo, G. Fourtakas, B.D. Rogers, J.L. Cercós-Pita, and R. Vacondio. Evaluation of reliability and efficiency of different boundary conditions in an sph code. In *Proceedings of the 10th International SPHERIC Workshop*, pages 45 – 52, Parma, Italy, June 16–18 2015.
- [39] J.M. Domínguez, A.J.C. Crespo, M. Gómez-Gesteira, and J.C. Marongiu. Neighbour lists in smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 67(12):2026–2042, 2011.

- [40] H.C. Edwards and D. Sunderland. Kokkos Array performance-portable many-core programming model. In *Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 1–10. ACM, 2012.
- [41] ESA. Slosat-FLEVO. http://web.archive.org/web/20050318162339/http://www.esa.int/export/esaMI/Launchers_Home/SEMNFZ0XDYD_0.html. [Online; accessed 17-October-2016].
- [42] M. Eswaran and G.R. Reddy. Liquid sloshing in fuel storage bays of advanced reactor subjected to earthquake loading. *International Conference on Vibration Problems 2015*, 144:1278 – 1285, 2016.
- [43] M.O. Faltinsen and A.N. Timokha. *Sloshing*. Cambridge University Press, 2009.
- [44] R. Fatehi and M. Manzari. A remedy for numerical oscillations in weakly compressible smoothed particle hydrodynamics. *Int. Journal for Numerical Methods in Fluids*, 67(9):1100–1114, 2011.
- [45] J. Feldman and J. Bonet. Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering*, 72(3):295–324, 2007.
- [46] M. Ferrand, D.R. Laurence, B.D. Rogers, D. Violeau, and C. Kassiotis. 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(4):446–472, 2012.
- [47] A. Ferrari, M. Dumbser, E.F. Toro, and A. Armanini. A new 3D parallel SPH scheme for free surface flows. *Computers & Fluids*, 38(6):1203– 1217, 2009.
- [48] Message Passing Interface Forum. Document for a standard message-passing interface. Technical report, Knoxville, TN, USA, 1994. www.netlib.org/utk/people/JD/JackDongarra/PAPERS/059_1994_mpi-a-message-passing-interface-standard.pdf [Online; accessed 17-October-2016].

- [49] G. Fourtakas. *Modelling multi-phase flows in Nuclear Decommissioning using SPH*. PhD thesis, The University of Manchester, 2014.
- [50] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 1977.
- [51] M. Gómez-Gesteira, B.D. Rogers, A.J.C. Crespo, R.A. Dalrymple, M. Narayanaswamy, and J.M. Domínguez. SPHysics - development of a free-surface fluid solver. Part 1: Theory and formulations. *Computers & Geosciences*, 48:289–299, 2012. SPHysics is available at: www.sphysics.org.
- [52] M. Gómez-Gesteira, B.D. Rogers, R.A. Dalrymple, and J.C. Crespo. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research*, 48(S1):6–27, 2010.
- [53] E.W. Graham and A.M. Rodriguez. The characteristics of fuel motion which affect airplane dynamics. Technical report, DTIC Document, 1951.
- [54] M. Greco. *A two-dimensional study of green-water loading*. PhD thesis, Fakultet for ingeniørvitenskap og teknologi, 2001.
- [55] S. Gross and A. Reusken. *Numerical methods for two-phase incompressible flow*. Springer, 2011.
- [56] Khronos Group. OpenCL. www.khronos.org/opencv/. [Online; accessed 17-October-2016].
- [57] X. Guo, B.D. Rogers, S. Lind, B. Tian, and M. Ashworth. Towards highly scalable incompressible smoothed particle hydrodynamics toolkit: optimization for real complex flow. In *27th International Conference on Parallel Computational Fluid Dynamics, May 2015, Montreal, Quebec, Canada*, 2015.
- [58] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International*, pages 63–70. SBC Petropolis, 2007.
- [59] F.H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

- [60] A. H  rault, G. Bilotta, and R.A. Dalrymple. SPH on GPU with CUDA. *Journal of Hydraulic Research*, 48(S1):74–79, 2010.
- [61] A. H  rault, G. Bilotta, and R.A. Dalrymple. Achieving the best accuracy in an SPH implementation. In *Proceedings of the 9th International SPHERIC Workshop*, pages 134–139, Paris, France, June 3–5 2014.
- [62] A. H  rault, G. Bilotta, R.A. Dalrymple, E. Rustico, and C. Del Negro. GPUSPH. www.gpusph.org. [Online; accessed 17-October-2016].
- [63] C.W. Hirt and B.D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.
- [64] J. Hongbin and D. Xin. On criteria for smoothed particle hydrodynamics kernels in stable field. *Journal of Computational Physics*, 202(2):699–709, 2005.
- [65] X.Y. Hu and N.A. Adams. Angular-momentum conservative smoothed particle dynamics for incompressible viscous flows. *Physics of Fluids (1994-present)*, 18(10):101702, 2006.
- [66] X.Y. Hu and N.A. Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.
- [67] X.Y. Hu and N.A. Adams. An incompressible multi-phase SPH method. *Journal of Computational Physics*, 227(1):264–278, 2007.
- [68] A.R. Ibrahim. *Liquid sloshing dynamics: theory and applications*. Cambridge University Press, 2005.
- [69] Flow Science Inc. Flow-3D. www.flow3d.com. [Online; accessed 17-October-2016].
- [70] R. Issa and D. Violeau. 3D schematic dam break and evolution of the free surface. <http://spheric-sph.org/tests/test-2>. [Online; accessed 17-October-2016].
- [71] J.H. Jeong, M.S. Jhon, J.S. Halowb, and J. van Osdol. Smoothed particle hydrodynamics: Applications to heat conduction. *Computer Physics Communications*, 153(1):71–84, 2003.
- [72] G. Joos and I.M. Freeman. *Theoretical physics*. Dover Publications, 1958.

- [73] A. Khayyer and H. Gotoh. Wave impact pressure calculations by improved SPH methods. *International Journal of Offshore and Polar Engineering*, 19(04), 2009.
- [74] Khronos. OpenGL. opengl.org/. [Online; accessed 17-October-2016].
- [75] S. Kulasegaram, J. Bonet, W.R. Lewis, and M. Profit. A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications. *Computational Mechanics*, 33(4):316–325, 2004.
- [76] J. Lacapere, B. Vieille, and B. Legrand. Experimental and numerical results of sloshing with cryogenic fluids. *Progress in Propulsion Physics*, 1:267–278, 2009.
- [77] C. Lada and A. Passaro. Experimental sloshing references test report. Technical report, ESA-ESTEC TEC-MPA Report, 2016.
- [78] R.T. Lahey. On the direct numerical simulation of two-phase flows. *Nuclear Engineering and Design*, 239(5):867–879, 2009.
- [79] A. Leroy, D. Violeau, A. Joly, and M. Ferrand. Buoyancy modelling with incompressible SPH and the unified semi-analytical wall boundary conditions. In *Proceedings of the 10th International SPHERIC Workshop*, pages 40 – 47, Parma, Italy, June 16–18 2015.
- [80] S.J. Lind, R. Xu, P.K. Stansby, and B.D. Rogers. 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(4):1499–1523, 2012.
- [81] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific, 2003.
- [82] M.B. Liu and G.R. Liu. Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of Computational Methods in Engineering*, 17(1):25–76, 2010.
- [83] M.B. Liu, G.R. Liu, and K.Y. Lam. Constructing smoothing functions in smoothed particle hydrodynamics with applications. *Journal of Computational and Applied Mathematics*, 155(2):263–284, 2003.

- [84] E.Y.M. Lo and S. Shao. Simulation of near-shore solitary wave mechanics by an incompressible SPH method. *Applied Ocean Research*, 24(5):275–286, 2002.
- [85] S.M. Longshaw and B.D. Rogers. Automotive fuel cell sloshing under temporally and spatially varying high acceleration using GPU-based smoothed particle hydrodynamics (SPH). *Advances in Engineering Software*, 83:31–44, 2015.
- [86] L.B. Lucy. A numerical approach to testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977.
- [87] D. Luebke. General-purpose computation on graphics hardware. In *SIG-GRAPH Workshop*, 2004.
- [88] S. Manenti, S. Sibilla, M. Gallati, G. Agate, and R. Guandalini. SPH simulation of sediment flushing induced by a rapid water flow. *Journal of Hydraulic Engineering*, 138(3):272–284, 2011.
- [89] J.C. Marongiu, F. Leboeuf, and E. Parkinson. 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(6):849–856, 2007.
- [90] S. Marrone, M. Antuono, A. Colagrossi, G. Colicchio, D. Le Touzé, and G. Graziani. δ -SPH model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*, 200(13):1526–1542, 2011.
- [91] S. Marrone, A. Colagrossi, D. Le Touzé, and G. Graziani. Fast free-surface detection and level-set function definition in SPH solvers. *Journal of Computational Physics*, 229(10):3652–3663, 2010.
- [92] A. Mayrhofer, M. Ferrand, C. Kassiotis, D. Violeau, and F.X. Morel. Unified semi-analytical wall boundary conditions in SPH: Analytical extension to 3D. *Numerical Algorithms*, 68(1):15–34, 2014.
- [93] A. Mayrhofer, B.D. Rogers, and D. Violeau. Investigation of wall bounded flows using SPH and the unified semi-analytical wall boundary conditions. *Computer Physics Communications*, 184(11):2515–2527, 2013.

- [94] S. McIntosh-Smith, T. Wilson, A.Á. Ibarra, J. Crisp, and R.B. Sessions. Benchmarking energy efficiency, power costs and carbon emissions on heterogeneous systems. *The Computer Journal*, 55(2):192–205, 2012.
- [95] A. Mokos, B.D. Rogers, P.K. Stansby, and J.M. Domínguez. Multi-phase SPH modelling of violent hydrodynamics on GPUs. *Computer Physics Communications*, 196:304–316, 2015.
- [96] D. Molteni and A. Colagrossi. A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. *Computer Physics Communications*, 180(6):861–872, 2009.
- [97] J.J. Monaghan. On the problem of penetration in particle methods. *Journal of Computational Physics*, 82(1):1–15, 1989.
- [98] J.J. Monaghan. Smoothed particle hydrodynamics. *Annual Review Astronomy and Astrophysics*, 30:543–574, 1992.
- [99] J.J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110(2):399–406, 1994.
- [100] J.J. Monaghan. SPH without a tensile instability. *Journal of Computational Physics*, 159(2):290–311, 2000.
- [101] J.J. Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review Fluid Dynamics*, 44:323–346, 2012.
- [102] J.J. Monaghan and J.B. Kajtar. SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180(10):1811–1820, 2009.
- [103] J.J. Monaghan and A. Kos. Solitary waves on a cretan beach. *Journal of waterway, port, coastal, and ocean engineering*, 125(3):145–155, 1999.
- [104] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33(3):333–353, 2000.
- [105] J.P. Morris, P.J. Fox, and Y. Zhu. Modeling low Reynolds number incompressible flows using SPH. *Journal of Computational Physics*, 136(1):214–226, 1997.
- [106] NASA. Modeling of slosh dynamics in cryogenic propellant tanks in microgravity environments. Technical report, 2008. NTRS Document ID: 20090022217 <https://ntrs.nasa.gov/>.

-
- [107] Netlib. BLAS. www.netlib.org/blas/. [Online; accessed 17-October-2016].
- [108] Numeca. FINE. www.numeca.com. [Online; accessed 17-October-2016].
- [109] Nvidia Corp. CUDA. www.nvidia.com/object/cuda_home_new.html. [Online; accessed 17-October-2016].
- [110] Nvidia Corp. CUDA C Programming Guide v7.5. docs.nvidia.com/cuda/cuda-c-programming-guide/. [Online; accessed 17-October-2016].
- [111] Nvidia Corp. Kepler whitepaper. www.nvidia.co.uk/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf. [Online; accessed 17-October-2016].
- [112] G. Oger, E. Jacquin, M. Doring, P.M. Guilcher, R. Dolbeau, P.L. Cabelguen, L. Bertaux, D. Le Touzé, and B. Alessandrini. Hybrid CPU-GPU acceleration of the 3D parallel code SPH-Flow. In *Proceedings of the 5th Spheric Workshop, Manchester, UK*, 2010.
- [113] H.A. Olsen and T. Hysing. A study of dynamic loads caused by liquid sloshing in LNG tanks. Technical report, Maritime Technical Information Facility, 1974.
- [114] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. Runge-Kutta method. In *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, pages 704–716. Cambridge University Press, 1992.
- [115] D.J. Price. Smoothed particle hydrodynamics: Things I wish my mother taught me. In *Advances in Computational Astrophysics: methods, tools and outcomes*, Cefalu, Italy, June 13–17 2011. [arxiv:1111.1259](https://arxiv.org/abs/1111.1259).
- [116] N.J. Quinlan, M. Basa, and M. Lastiwka. Truncation error in mesh-free particle methods. *International Journal for Numerical Methods in Engineering*, 66(13):2064–2085, 2006.
- [117] A. Rafiee, F. Pistani, and K. Thiagarajan. Study of liquid sloshing: numerical and experimental approach. *Computational Mechanics*, 47(1):65–75, 2011.
- [118] C. Raskin, N. Frontiere, and J. M. Owen. CRKSPH - a conservative reproducing kernel smoothed particle hydrodynamics scheme. In *Proceedings of the 10th*

- International SPHERIC Workshop*, pages 45 – 52, Parma, Italy, June 16–18 2015.
- [119] C. Raskin and J.M. Owen. Examining the accuracy of astrophysical disk simulations with a generalized hydrodynamical test problem. *arXiv preprint arXiv:1607.04293*, 2016.
- [120] S. Rebouillat and D. Liksonov. Fluid-structure interaction in partially filled liquid containers: A comparative review of numerical approaches. *Computers and Fluids*, 39(5):739–746, 2010.
- [121] D. Reed, J. Yu, H. Yeh, and S. Gardarsson. Investigation of tuned liquid dampers under large amplitude excitation. *Journal of Engineering Mechanics*, 124(4):405–413, 1998.
- [122] Y. Renardy and M. Renardy. PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method,. *Journal of Computational Physics*, 183(2):400–421, 2002.
- [123] C. Rhie and W. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, 1983.
- [124] W.J. Rider and D.B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112–152, 1998.
- [125] B.D. Rogers and R.A. Dalrymple. SPH modeling of breaking waves. In *Proceedings of the 29th International Conference on Coastal Engineering*, pages 415–427, Lisbon, Portugal, 2004.
- [126] B.D. Rogers, R.A. Dalrymple, M. Gómez-Gesteira, and A.J.C. Crespo. User Guide for the parallelSPHysics Code using MPI v2.0. www.sphysics.org.
- [127] B.D. Rogers, R.A. Dalrymple, and A. Robert. SPH modeling of tsunami waves. *Advanced numerical models for simulating tsunami waves and runup*, 10:75–100, 2008.
- [128] A. Royon-Lebeaud, E.J. Hopfinger, and A. Cartellier. Liquid sloshing and wave breaking in circular and square-base cylindrical containers. *Journal of Fluid Mechanics*, 577:467–494, 2007.

- [129] J.A. Salzman and W.J. Masica. *Lateral sloshing in cylinders under low-gravity conditions*. National Aeronautics and Space Administration, 1969.
- [130] R. Scardovelli and S. Zaleski. Direct numerical simulation of free surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31(1):567–603, 1999.
- [131] P. Schallhorn, J. Roth, B. Marsell, D. Kirk, H. Gutierrez, A. Saenz-Otero, D. Dorney, and J. Moder. Acquisition of long-duration, low-gravity slosh data utilizing existing ISS equipment (SPHERES) for calibration of CFD models of coupled fluid-vehicle behavior. In *1st Annual International Space Station (ISS) Research and Development Conference*, Denver, CO, United States, June 26–28 2012.
- [132] B. Serván-Camas, J.L. Cercós-Pita, J. Colom-Cobb, J. García-Espinosa, and A. Souto-Iglesias. Time domain simulation of coupled sloshing–seakeeping problems by SPH–FEM coupling. *Ocean Engineering*, 123:383–396, 2016.
- [133] J.R. Shao, H.Q. Li, G.R. Liu, and M.B. Liu. An improved SPH method for modeling liquid sloshing dynamics. *Computers & Structures*, 100:18–26, 2012.
- [134] A. Souto-Iglesias, E. Botia-Vera, and G. Bulian. Repeatability and two-dimensionality of model scale sloshing impacts. In *The 22nd International Offshore and Polar Engineering Conference*, Rhodes, Greece, June 17–22 2012. International Society of Offshore and Polar Engineers.
- [135] A. Souto-Iglesias, L. Delorme, L. Pérez-Rojas, and S. Abril-Pérez. Liquid moment amplitude assessment in sloshing type problems with smooth particle hydrodynamics. *Ocean Engineering*, 33(11-12):1462–1484, 2006.
- [136] SPHERIC. SPH European Research Interest Community. spheric-sph.org/. [Online; accessed 17-October-2016].
- [137] M. Sussman, E. Fatemi, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flows. *Computers & Fluids*, 27(5):663–680, 1998.
- [138] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

- [139] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobbs journal*, 30(3):202–210, 2005.
- [140] A.M. Tartakovsky, P. Meakin, T.D. Scheibe, and R.M. Eichler West. Simulations of reactive transport and precipitation with smoothed particle hydrodynamics. *Journal of Computational Physics*, 222(2):654–672, 2007.
- [141] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct numerical simulations of gas-liquid multiphase flows*. Cambridge University Press, 2011.
- [142] S.O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.
- [143] E. Uzgoren, R. Singh, J. Sim, and W. Shyy. Computational modeling for multiphase flows with spacecraft application. *Progress in Aerospace Sciences*, 43(4):138–192, 2007.
- [144] G.L. Vaughan, T.R. Healy, K.R. Bryan, A.D. Sneyd, and R.M. Gorman. Completeness, conservation and error in SPH for fluids. *International journal for numerical methods in fluids*, 56(1):37–62, 2008.
- [145] D. Violeau. Dissipative forces for Lagrangian models in computational fluid dynamics and application to smoothed-particle hydrodynamics. *Physical Review E*, 80(3):036705, 2009.
- [146] D. Violeau. *Fluid mechanics and the SPH method: Theory and applications*. Oxford University Press, 2012.
- [147] D. Violeau and B.D. Rogers. Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016.
- [148] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995.
- [149] M. Wörmer. Numerical modeling of multiphase flows in microfluidics and micro process engineering: A review of methods and applications. *Microfluid and Nanofluid*, 12(6):841–886, 2012.

-
- [150] R. Xu, P. Stansby, and D. Laurence. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725, 2009.
- [151] M. Yildiz, R.A. Rook, and A. Suleman. SPH with the multiple boundary tangent method. *International Journal for Numerical Methods in Engineering*, 77(10):1416–1438, 2009.