Federal University of Pernambuco

Center of Informatics

Computer Science Graduation

On Turbulence Coupled SPH for Liquid Sloshing Simulation

Graduation Project

Daniel de Souza Queiroga

Adviser: Silvio de Barros Melo
Co-adviser: Caio Jose dos Santos Brito (Voxar Labs/CIn-UFPE)

July 2018

Daniel de Souza Queiroga

On Turbulence Coupled SPH for Liquid Sloshing Simulation

Thesis submitted to the Center of Informatics of the Federal University of Pernambuco as partial requirement for the degree of Bachelor of Science in Computer Science

Federal University of Pernambuco
Center of Informatics
Computer Science Graduation

Adviser: Silvio de Barros Melo
Co-adviser: Caio Jose dos Santos Brito (Voxar Labs/CIn-UFPE)

Resumo

Simulação de fluídos é uma área com diversas aplicações, de filmes e jogos até construção naval e defesa costeira. Podendo ser separada entre simulações de malhas e de partículas,os avanços em simulações de fluídos trazem resultados cada vez melhores. Um caso de grande interesse para esse tipo de simulação é o estudo do fenômeno de *sloshing*, que ocorre em tanques carregando líquidos, como em tanques de combustíveis e em contêineres presentes em navios. Neste trabalho faremos comparações entre soluções baseadas no método de partículas SPH combinado com modelos de turbulência, visando uma simulação realista de fluxos.

Palavras-chave: *Simulação de fluídos, SPH, Sloshing, Turbulência.*

Abstract

Fluid simulation is an area with various applications, ranging from movies and games, to shipbuilding and coastal defense. Being divided in mesh-based and meshless methods, improvements in fluid simulation achieve ever better results. A case of great interest is the study of the liquid sloshing phenomenon, which occurs in liquid inside other objects, such as fuel tanks and containers in ships. In this work we will make comparisons between solutions that use the SPH particle method coupled with turbulence models, aiming at realistic flow simulation.

Keywords: *Fluid Simulation, SPH, Sloshing, Turbulence.*

Contents

Chapter 1

Introduction

1.1 Motivation

A wide range of applications use fluid simulation: games, movies and engineering being the biggest areas where it is applied. By making compromises between physical accuracy and visual quality, creators can generate plausible results for entertainment while reducing time and processing costs, even being able to develop real-time solutions[1][2].

Fluid simulation can be separated in mesh based solutions and mesh-free solutions. The former makes use of a grid of cells to represent the fluid[3], this approach usually achieves more efficient results, with higher numerical accuracy[4]. But achieving physical accuracy is not a trivial feat and the complexity of the implementation can become a hurdle[5], as poor management of the grid can introduce errors[4] and lead to non realistic behaviors.

Mesh-free, or particle based, approaches rely on a set of particles to represent the fluid, boundaries and interacting objects[6]. Each particle stores parameters that are used to calculate values as pressure, density, velocity, among others[5]. The particles motion is given by equations that use those parameters as input. Because of this, when compared to grid-based solutions, the implementation becomes more straightforward, while maintaining good physical accuracy.

In this work, we will be using the Smoothed Particle Hydrodynamics (SPH) method, which was originally developed to solve three-dimensional astrophysical problems[7]. The SPH Method consists of three main steps: neighborhood search, computation of each particle acceleration and time integration. The first step can be the most time consuming if dealt with in a poor manner, such as through brute-force. To circumvent this it is usual to implement spatial access structures, like grids[8] or an octree[9]. The forces considered in the simulation are gravity, pressure and viscosity[10]. The second step usually is solved using the Navier-Stokes

equations[10] but the pressure can also be determined by the Poisson equation[10], though the latter can make the simulation slower as it creates the need to solve a sparse linear system. The third step can be done in a few different ways, such as locally and globally[11], and with fixed or varying time-steps[12].

After its creation, SPH has been used to solve a plethora of problems involving fluid simulation and other problems, ranging from coastal defense[13] and tsunami and landslide prediction[14] to surface erosion[15] and liquid sloshing[16].

In this work we decided to focus on liquid sloshing, given the importance of understanding this phenomenon for real world applications. Sloshing is the movement of liquids inside objects under motion when there is a free surface[17]. A common example found in the literature is fuel tanks on oil tankers[18]. But other examples found in the literature study sloshing occurrence in eggs[19], beer glasses[20], containers in moving trucks[21], and more.

Coupling with turbulence models can lead to results with better inner fluid motion and vortexes representation.

1.2 Objective

The main objective of this graduation work is to explore the liquid sloshing phenomenon using the Smoothed Particle Hydrodynamics method for turbulent fluid simulations. In order to accomplish our objective, we intend to implement different turbulence models proposed in the literature and make comparisons between them and available experimental data. Existing SPH implementations will be chosen based on their accuracy and extensibility, and the proposed turbulence models will then be extended to the implementations

Although validations exist in the literature, the SPH method is still not considered as mature as other methods, such as the Finite Element Method. For this reason, we believe that the proposed tests can be used not only to study the phenomenon, but also to further validate the use of SPH for realistic sloshing simulation, this way creating a better base for future works and validations.

1.3 Contributions

The contributions of this work are the following:

- Validation for sloshing simulations using the DualSPHysics[22] solver for both the SPS and RANS model.
- Comparisons between CPU and GPU numerical accuracy and run time.
- A method to compute the wave height relative to the moving boundary

1.4 Document overview

In the second chapter, the basic SPH formulation is explained, building up on this explanation we explain the turbulence models and time integration schemes, thus giving an understanding of the fundamental theoretical process of the work. On the third chapter, the methodology is explained, showing the simulation framework and the method devised for computing the wave height. On the fourth chapter, the results are shown and are briefly analyzed. Then, on the fifth chapter a brief conclusion is made and after that some ideas for future works are elicited.

Chapter 2

Smoothed Particle Hydrodynamics

2.1 Basic formulation

Smoothed Particles Hydrodynamics (SPH) is a meshless method that uses discrete Navier-Stokes equations to apply forces at each particle, according to the properties of neighboring particles. For each particle, a function that takes into account the distance between particles determines the set of neighboring particles and a factor for the forces applied by each particle. The distance is usually called the *smoothing length*, denoted as h. For each simulation step, new physical quantities are calculated for all particles and their positions are updated accordingly.

As we need a discrete model for the simulation, first the conservation laws of continuum fluid dynamics need to be rewritten as an integral equation based on an interpolation (or weighing) function, commonly referred as the *kernel function*, denoted as W on the following equations. The forces that act at position $\boldsymbol{r}$ from other positions $\boldsymbol{r'}$ will be given by the function $F(\boldsymbol{r})$:

$$F(\boldsymbol{r}) = \int F(\boldsymbol{r'})W(\boldsymbol{r} - \boldsymbol{r'}, h)d\boldsymbol{r'} \tag{2.1}$$

Eq.(2.1) can be further be transformed into a discrete form, suitable to work with a set of particles instead of positions $F(\boldsymbol{r}_a)$. This approximated form is given by:

$$F(\boldsymbol{r}_a) \approx \sum_b F(\boldsymbol{r}_b)W(\boldsymbol{r}_a - \boldsymbol{r}_b, h)\Delta v_b \tag{2.2}$$

Where $\boldsymbol{r}_a$ is the position of the particle for which we want to calculate the acting forces, $\boldsymbol{r}_b$ represents the position of a neighboring particle and Δv_b is the

volume of the neighboring particle. As the volume is calculated as $\Delta v_b = \frac{m_b}{\rho_b}$ from the mass m_b and density ρ_b associated with the particle b, Eq.(2.2) can be rewritten as:

$$F(\boldsymbol{r}_a) \approx \sum_b F(\boldsymbol{r}_b)\frac{m_b}{\rho_b}W(\boldsymbol{r}_a - \boldsymbol{r}_b, h) \tag{2.3}$$

The particle's mass is a constant, but its density ρ is calculated each time step by the following equation, called the continuity equation:

$$\frac{d\rho_a}{dt} = \sum_b m_b \boldsymbol{v}_{ab} \cdot \nabla_a W_{ab} \tag{2.4}$$

The final model has a term for pressure $-\frac{1}{\rho}\nabla P$, gravitational force $\boldsymbol{g}$ and dissipative terms $\boldsymbol{\Gamma}$. It is commonly called the momentum conservation equation with which we update the particle velocity $\boldsymbol{v}$ for each time step t:

$$\frac{d\boldsymbol{v}}{dt} = -\frac{1}{\rho}\nabla P + \boldsymbol{g} + \boldsymbol{\Gamma} \tag{2.5}$$

Simpler simulations may ignore $\boldsymbol{\Gamma}$, in this work aside $\boldsymbol{\Gamma}$, an extra turbulence term is added.

2.2 The equation of state

The chosen solver[22] uses weakly compressible equation of state to solve the pressure value in Eq.(2.5) which is given by[23]:

$$P = b\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right) \tag{2.6}$$

Where $\gamma = 7$, $b = \frac{c_0^2\rho_0}{\gamma}$, the reference density is $\rho_0 = 1000kg/m^3$ and the speed of sound at the reference density is c_0.

For comparison purposes, an equation for incompressible flow is implemented, given by[16]:

$$P = c_0^2\rho \tag{2.7}$$

2.3 The kernel function

The kernel function W is used to determine the neighboring particles b that interact with particle a at each simulation step. Three important points on kernel

functions will be briefly summarized here, but for a deeper explanation we recommend reading the overview by Liu et al[5].

1. **Positivity:** The function should not return negative results. In case this is not true unphysical behaviors can be a consequence.
2. **Decay:** The further particle b is from particle a, the less impact its forces will have in final calculation.
3. **Symmetric property:** The function should be symmetric. Neighboring particles b located at different positions but with the same distance from a should have the the same factor for the forces calculation.

In this work we will be using the cubic spline kernel[24]:

$$W(r,h) = \alpha_D \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \leq q \leq 1, \\ \frac{1}{4}(2-q)^3 & 1 \leq q \leq 2, \\ 0 & q \geq 2 \end{cases} \tag{2.8}$$

and the quintic kernel[25]:

$$W(r,h) = \alpha_D \left(1 - \frac{q}{2}\right)^4 (2q+1) \qquad 0 \leq q \leq 2 \tag{2.9}$$

For both kernels, q is given by $q = \frac{r}{h}$ where r is the distance between particles a and b and h is the smoothing length. On the cubic spline kernel, α_D is valued as $\frac{10}{7\pi h^2}$ in 2-D simulations and $\frac{21}{16\pi h^3}$ for 3-D cases.

2.4 Viscosity term

Going back to Eq.(2.5), $\mathbf{\Gamma}$ is where we will introduce the viscosity term that can be separated in two types: artificial viscosity and laminar viscosity. In this work, the turbulence model is coupled with the laminar viscosity term, and for validation purposes, both the artificial viscosity and the laminar viscosity models are used.

2.4.1 Artificial viscosity

In this work, the artificial viscosity term Π_{ab} is defined as proposed by Monaghan[26]:

$$\Pi_{ab} = \begin{cases} \overline{\frac{-\alpha c_{ab}\mu_{ab}}{\rho_{ab}}} & \boldsymbol{v}_{ab} \cdot \boldsymbol{r}_{ab} < 0, \\ 0 & \boldsymbol{v}_{ab} \cdot \boldsymbol{r}_{ab} > 0 \end{cases} \tag{2.10}$$

Where $\rho_{ab} = \rho_a - \rho_b$ is the difference in density between a and b, $\boldsymbol{v}_{ab} = \boldsymbol{v}_a - \boldsymbol{v}_b$ is the difference in velocity between a and b and $\boldsymbol{r}_{ab} = \boldsymbol{r}_a - \boldsymbol{r}_b$ is the distance between a and b. $\mu_{ab} = \frac{h\boldsymbol{v}_{ab}\cdot\boldsymbol{r}_{ab}}{\boldsymbol{r}_{ab}^2+\eta^2}$ with $\eta^2 = 0.01h^2$ and $\overline{c_{ab}} = 0.5(c_a + c_b)$ being the mean speed of sound. α is a user defined coefficient that can be tuned to achieve proper dissipation.

The momentum conservation equation (Eq.(2.5)) can then be rewritten for a given particle a as:

$$\frac{d\boldsymbol{v}_a}{dt} = -\sum_b m_b\left(\frac{P_b + P_a}{\rho_b \cdot \rho_a} + \Pi_{ab}\right)\nabla_a W_{ab} + g \tag{2.11}$$

2.4.2 Laminar viscosity

The laminar viscous stress $v_o\nabla^2\boldsymbol{v}$ for a given particle a can be written as[27]:

$$\left(v_o\nabla^2\boldsymbol{v}\right)_a = \sum_b m_b\left(\frac{4v_o\boldsymbol{r}_{ab}\cdot\nabla_a W_{ab}}{(\rho_a+\rho_b)(\boldsymbol{r}_{ab}^2+\eta^2)}\right)\boldsymbol{v}_{ab} \tag{2.12}$$

where v_o is the kinematic viscosity (for water, usually $10^{-6}m^2s$). Substituting $\boldsymbol{\Gamma}$ with $v_o\nabla^2\boldsymbol{v}$, we can rewrite Eq.(2.5) for a given particle a as:

$$\frac{d\boldsymbol{v}_a}{dt} = -\sum_b m_b\left(\frac{P_b + P_a}{\rho_b \cdot \rho_a}\right)\nabla_a W_{ab} + \boldsymbol{g} + \sum_b m_b\left(\frac{4v_o\boldsymbol{r}_{ab}\cdot\nabla_a W_{ab}}{(\rho_a+\rho_b)(\boldsymbol{r}_{ab}^2+\eta^2)}\right)\boldsymbol{v}_{ab} \tag{2.13}$$

2.5 Turbulence models

In the following equations, the term $\frac{1}{\rho}\nabla\cdot\vec{\tau}$ represents the turbulence model. Two turbulence models are used in this work: the Sub-Particle Scale (SPS) model and the Reynolds Averaged Navier-Stokes (RANS) model.

Before explaining the models, we will rewrite Eq.(2.5) to better visualize the following steps:

$$\frac{d\boldsymbol{v}}{dt} = -\frac{1}{\rho}\nabla P + \boldsymbol{g} + v_o\nabla^2\boldsymbol{v} + \frac{1}{\rho}\nabla\cdot\ \vec{\tau} \tag{2.14}$$

The SPS model implementation is already available in the DualSPHysics solver. The RANS model was chosen for comparison because it does not require complex modifications to the SPS model. We intend to compare the numerical results and also the overall time each model requires. By using the RANS model we hope to achieve results comparable to the SPS model, but faster, as this model is a little simpler.

2.5.1 The RANS model

In the RANS model, $\vec{\tau}$ in eq.(2.14) is rewritten as:

$$\vec{\tau_{ab}} = \overline{\rho}\Big(2v_t S_{ab} - \frac{2}{3}k\delta_{ab}\Big) \tag{2.15}$$

where v_t is the turbulence eddy viscosity given by Eq.(2.16) and S_{ab} is given by Eq.(2.17). k is the kinetic energy, $l = C_s h$ is the maxing length, where h is the smoothing length and $C_s = 0.12$ is the Smagorinsky constant.

$$v_t = l^2\sqrt{2S_{ab}S_{ab}} \tag{2.16}$$

$$S_{ab} = \frac{1}{2}\left(\frac{\delta \boldsymbol{v}_a}{\delta \boldsymbol{r}_b} + \frac{\delta \boldsymbol{v}_a}{\delta \boldsymbol{r}_b}\right) \tag{2.17}$$

The discrete form of $\frac{1}{\rho}\nabla \cdot \vec{\tau}$ is given by[27]:

$$\frac{1}{\rho}\nabla \cdot \ \vec{\tau} = \sum_b m_b\left(\frac{\vec{\tau_a}}{\rho_a} + \frac{\vec{\tau_b}}{\rho_b}\right)\nabla_a W_{ab} \tag{2.18}$$

2.5.2 The SPS model

The SPS model was firts proposed by Gotoh et al[28], for their Moving Particle Semi-implicit (MPS) simulation. In this work the definition proposed by Dalrymple et al[29] is used. The main difference between the RANS model and the SPS model used in this work is in the $\vec{\tau}$ term. The term is rewritten as, where $C_I = 0.00066$ as defined by Blin et al[30]:

$$\vec{\tau_{ab}} = \overline{\rho}\Big(2v_t S_{ab} - \frac{2}{3}k\delta_{ab}\Big) - \frac{2}{3}\overline{\rho}C_I\Delta^2\delta_{ab} \tag{2.19}$$

As the SPS model has this added term, we believe that by using the RANS model we will be able to achieve faster results without a great impact on the numerical accuracy.

2.6 Time integration

The chosen solver[22] has two options for time integration, the Verlet scheme[31] and the Symplectic scheme, proposed by Leimkuhler et al [32]. Due to the two stage nature of the Symplectic scheme, it yields better numerical results and was chosen for this work.

2.6.1 Symplectic scheme

The Symplectic scheme used in the solver has an accuracy in time of $O(\Delta t^2)$. It has two stages, the predictor and the corrector stages.

During the predictor stage, the position and density are estimated for the middle of the time step, according to:

$$\boldsymbol{r}_a^{n+\frac{1}{2}} = \boldsymbol{r}_a^n + \frac{\Delta t}{2}\boldsymbol{v}_a^n \tag{2.20}$$

$$\rho_a^{n+\frac{1}{2}} = \rho_a^n + \frac{\Delta t}{2}D_a^n \tag{2.21}$$

Then, at the corrector stage, $\frac{d\boldsymbol{v}_a^{n+\frac{1}{2}}}{dt}$ is used to calculate the correct velocity and position, according to:

$$\boldsymbol{v}_a^{n+1} = \boldsymbol{v}_a^{n+\frac{1}{2}} + \frac{\Delta t}{2}\boldsymbol{F}_a^{n+\frac{1}{2}} \tag{2.22}$$

$$\boldsymbol{r}_a^{n+1} = \boldsymbol{r}_a^{n+\frac{1}{2}} + \frac{\Delta t}{2}\boldsymbol{v}_a^{n+1} \tag{2.23}$$

Finally the correct density value $\frac{d\rho_a^{n+1}}{dt} = D_a^{n+1}$ is calculated using $\boldsymbol{v}_a^{n+1}$ and $\boldsymbol{r}_a^{n+1}$.

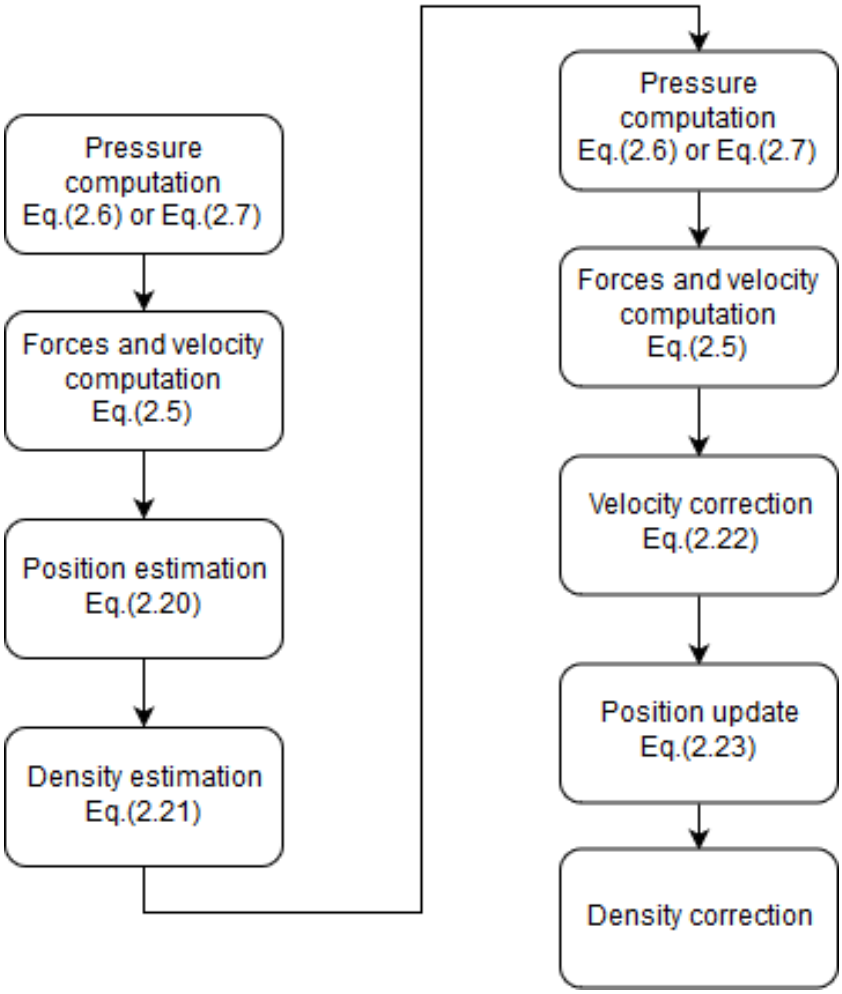


Figure 2.1: Diagram for one simulation step using the Symplectic scheme.

Chapter 3

Methodology

3.1 Simulation setup and computer configuration

Two scenario variations were simulated, each scenario with a baffle and no baffle case. For all of the simulations the tank dimensions according to Fig.(3.1) were height $t_h = 1.15m$ and length $t_l = 1.73m$. In both scenarios the tank is moving in a sinusoidal fashion ($S(t) = A \cdot cos(\frac{2\pi t}{T})$), with an amplitude $A = 0.025m$. For the first variation, the water height was $w_h = 0.6m$ and the period of the motion was $T = 1.3s$ and for the second variation $w_h = 0.5m$ and $T = 1.875s$. The baffles in both variations have height $b_h = 0.3m$ and length $b_l = 0.025m$. For the artificial viscosity an viscosity value of $\alpha = 0.01$ was chosen, given its performance in other simulations[13]. For the turbulence models, the kinetic viscosity value was set to $v_o = 10^{-6}m^2s$, for water simulation.

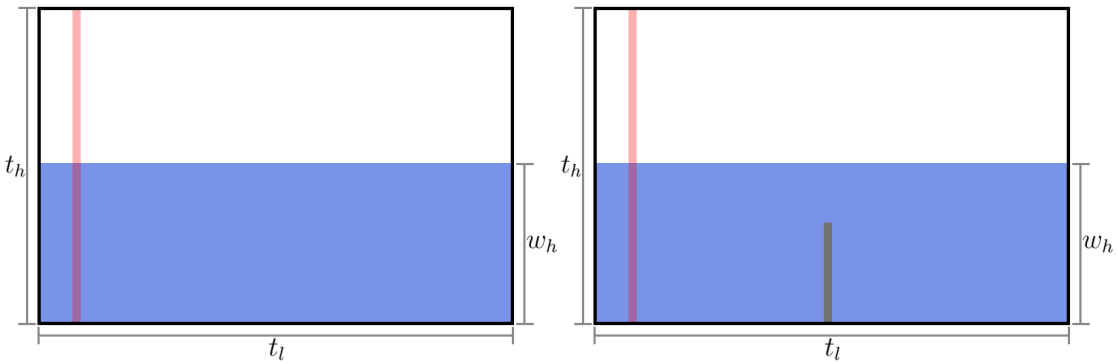


Figure 3.1: Simulation scenario diagram. t_h is the tank height, t_l tank length and w_h initial water height. The red zone indicates the position along t_l where the wave height is computed. Left scenario with baffle, right scenario without baffle

The specifications of the computer used for the simulations are summarized in Table 3.1.

Part	Memory	Memory clock rate	memory bus width	Processing clock rate	Number of cores
Intel i5-4590	8 GB (DDR3)	1600 MHz	64-bit	3.3 GHz	4
GeForce GTX 1060	6 GB (DDR5)	4004 MHZ	192-bit	1.78 GHz	1280 (CUDA)

Table 3.1: Computer specification

3.2 DualSPHysics and simulation framework

As the focus of this work was to study the sloshing phenomenon and not implement an SPH solver from the ground-up, we chose to use the DualSPHysics[22] solver. This choice was made because by doing so, we were able to focus only on the modifications needed to implement the RANS model and the incompressible equation of state.

The DualSPHysics solver is an open source solver developed by Universidade de Vigo and the University of Manchester. For its open source nature, this solver has great extensibility and easily available documentation. It has also been validated for various simulation scenarios in the literature, including dam break simulation[33], wave generation and propagation[13], fluid and rigid body interaction[34] and more.

The simulations ran on a CUDA[35] enabled GPU, with CUDA v9.1, utilizing the Symplectic time integration scheme.

The simulation scenarios are modeled after the experiment made by Faltinsen et al[36] and simulations by Shao et al[16].

3.3 Wave height

One problem encountered during the implementation process was that, with the available tools, the fluid height could only be batch computed for given points at fix positions relative to the whole simulation space. But in our simulations the tank is always moving. Besides that, the shear amount of parts each simulation produced made manually finding the height information prohibitive. For this reasons, a method for calculating the wave height relative to the moving tank was devised. We will now briefly explain it.

The main idea is to define an initial x position to be used as a reference. For each step a new x position X_{step} is calculated according to the motion of the tank and the wave height for each step is thus computed relative to the moving tank and not to the whole simulation space.

The algorithm receives as input the desired position $X_{desired}$, the boundary and fluid data for the simulation $Bounds$ and $Fluids$ and a reference density value ρ_{ref} used to find surface particles. $X_{desired}$ is used to find the closest boundary particle to that x position and saves that particle's ID as $bound_{ID}$.

From the simulation we have each particle's position for each step, there is no need to actually calculate $step_x$, we simply need to retrieve this information with $bound_{ID}$ we saved before.

After X_{step} is determined, we need to find the wave height. To do that we first filter the fluid particle by density, to find the particles closest to the surface. Then the surface particles are filtered yet again, this time according to their x position, returning only those particles that are sufficiently close to X_{step}. As of the time of writing, from this last filtering, the highest z position is chosen as the wave height.

The algorithm is summarized in pseudo-code below, and some remarks are made after.

Algorithm 1 Motion relative wave height

procedure RELATIVE HEIGHT($X_{desired}, Bounds, Fluids, \rho_{ref}$)
    $j \leftarrow 0$
    $bound_{ID} \leftarrow$ CLOSESTBOUNDARYPARTICLE($Bounds_j, X_{desired}$)
    for each simulation step **do**
        $X_{step} \leftarrow$ GETREFERENCEX($Bounds_j, bound_{ID}$)
        $surface \leftarrow$ FILTERSURFACE($Fluids_j, \rho_{ref}$)
        $closestSurfaceParticles \leftarrow$ FILTERALONGX($surface, X_{step}$)
        $heightValues_j \leftarrow max(closestSurfaceParticles_z)$
        $j \leftarrow j + 1$
    end for
end procedure

Although this method works, some points can be improved, mainly the surface reconstruction. As it is, low resolution simulations can lead to not so smooth results. Other point for improvement is that the list of surface points from one step to the other, probably, does not change a lot. This gives room for optimizing the surface reconstruction by only looking for particles around a given area.

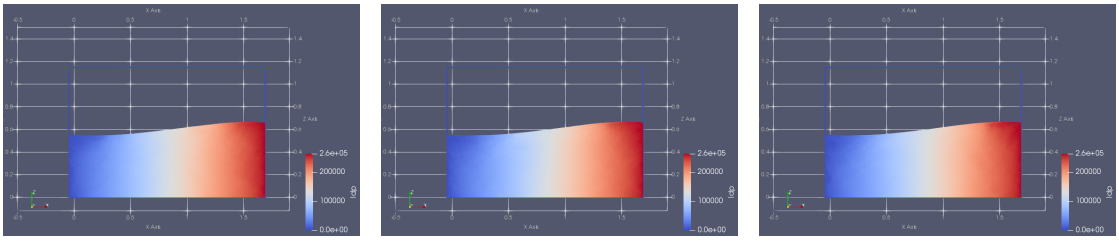
Chapter 4

Results

4.1 Visual results

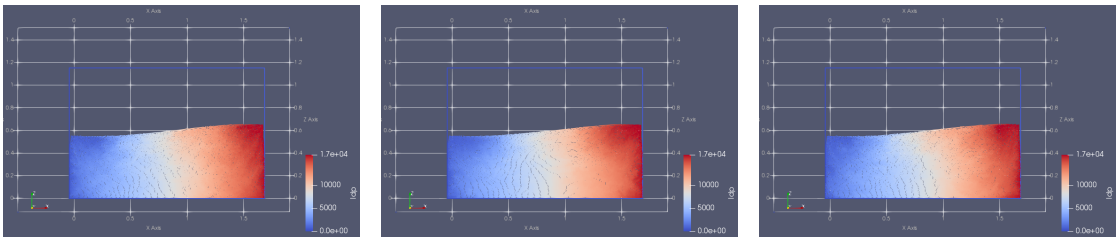
For variation 1, all the 3 models performed similarly visually. The particle displacement along time on the inside of the fluid was more uniform when the turbulence models were used for variation 1. But this did not affect the results greatly.

Even though the simulations visual results were similar for particle distribution, as can be seen on Fig.(4.1) through Fig.(4.2), when we compare the velocities we can see that without turbulence models there is some dampening Fig(4.3).



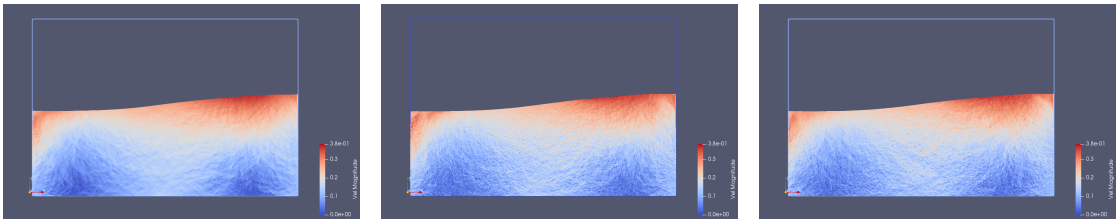
No model SPS RANS

Figure 4.1: $dp = 0.002$ m, 261218 particles.



No model SPS RANS

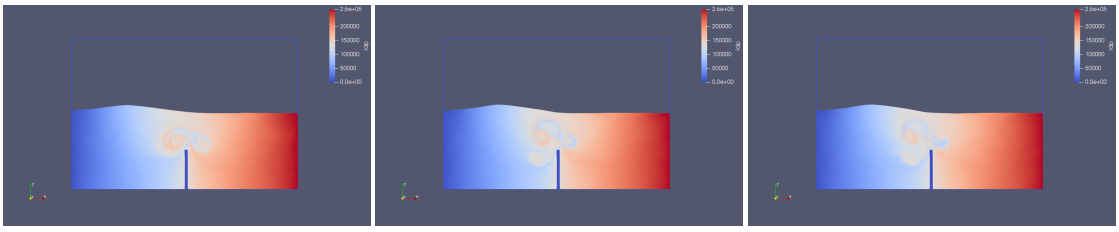
Figure 4.2: $dp = 0.008$ m, 16632 particles.



No model SPS RANS

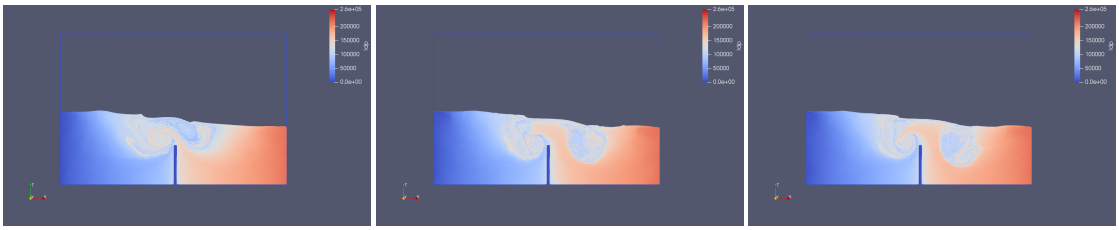
Figure 4.3: Velocity profiles for variation 1. We are able to see some dampening when no model is used.

As can be seen on Fig.(4.4) and Fig.(4.5), although the surface level is similar for the three models, the simulations without a turbulence model shows some dampening in the internal vortexes. As a middle baffle is introduced in variation 2, the flow becomes more violent, resulting in more violent vortexes and the dampening becomes clear.



No model SPS RANS

Figure 4.4: Baffle comparisons for variation 1, $t = 4$ s, 261218 particles.



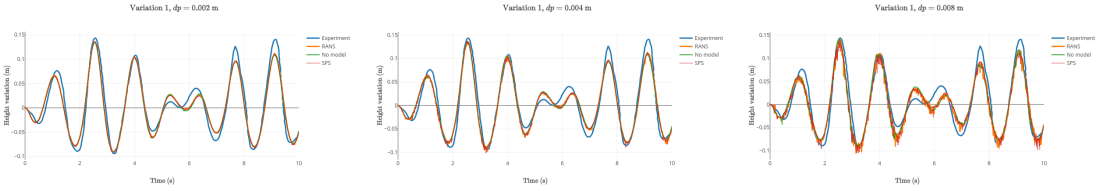
No model SPS RANS

Figure 4.5: Baffle comparisons for variation 2, $t = 7$ s, 218018 particles.

4.2 Numerical results

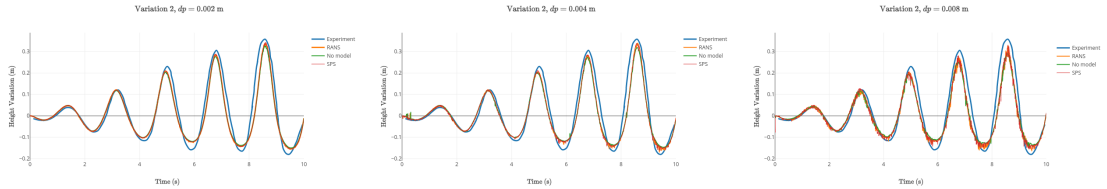
Using both the incompressible and weakly compressible state equation, all simulations yielded similar results for wave height variation, with good agreement to the experimental results. The dissimilarity between the SPS and RANS models was 0.001, computed using the Kolmogorov-Smirnov test[37]. With this small difference we achieved the intended goal of comparable results between the two models.

Lowering the resolution leads to impaired wave height reading Fig.(4.6) and Fig.(4.7). This further indicates that the solution proposed in Alg.(1) can be improved with better surface tracking. Besides this, the simulations were in good accordance with the experiment, thus validating the models used Fig(4.6) and Fig.(4.7).



$dp = 0.002$ m $dp = 0.004$ m $dp = 0.008$ m

Figure 4.6: Graphs for variation 1

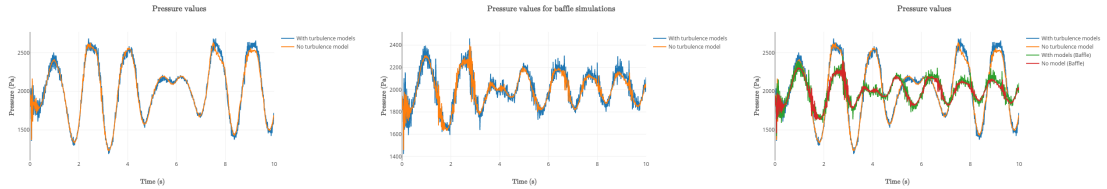


$dp = 0.002$ m $dp = 0.004$ m $dp = 0.008$ m

Figure 4.7: Graphs for variation 2

The pressure at the left wall was also analyzed, with a probe placed 0.4 m up from the bottom of the tank. Even though the surface level was very close for all the 3 models, analyzing the pressure profile we can see that there is also some dampening when no model is used, as expected from the visual velocity results.

Also as expected from the visual results and height variation results, the baffle variation exhibits lower overall pressure values. It can also be noted that initially the pressure values for the scenarios with and without baffles are similar, but when the height reduction from the baffles becomes more apparent, the same happens for the pressure reduction on the wall. As both turbulence models behaved similarly, they are shown together in Fig(4.8).



No baffle With baffle Baffle and no baffle

Figure 4.8: Graphs for pressure values.

4.3 Performance results

In Fig.(4.9) we have a breakdown of the simulation time, roughly based around the diagram shown on Fig.(2.1). We can see that the biggest part of the simulation is spent on the forces computations, being over 63% of the simulation time. This shows that this step is a great candidate for future optimizations.

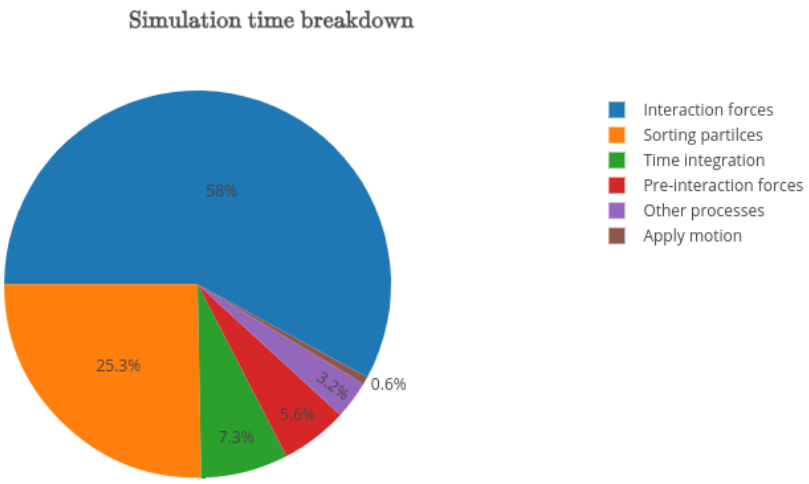


Figure 4.9: GPU simulation breakdown.

For comparison purposes, simulations using the CPU were also made. Simulations with 116548, 510601 and 1041433 particles were run multiple times to get an average time. The simulation results were similar, with a difference of 0.002, using the Kolmogorov-Smirnov test[37]. But the CPU simulations were up to 21 times slower when compared with one that used the GPU, and, as can be seen on Fig.(4.10), the higher the particle count the bigger the difference between CPU and GPU run time became.

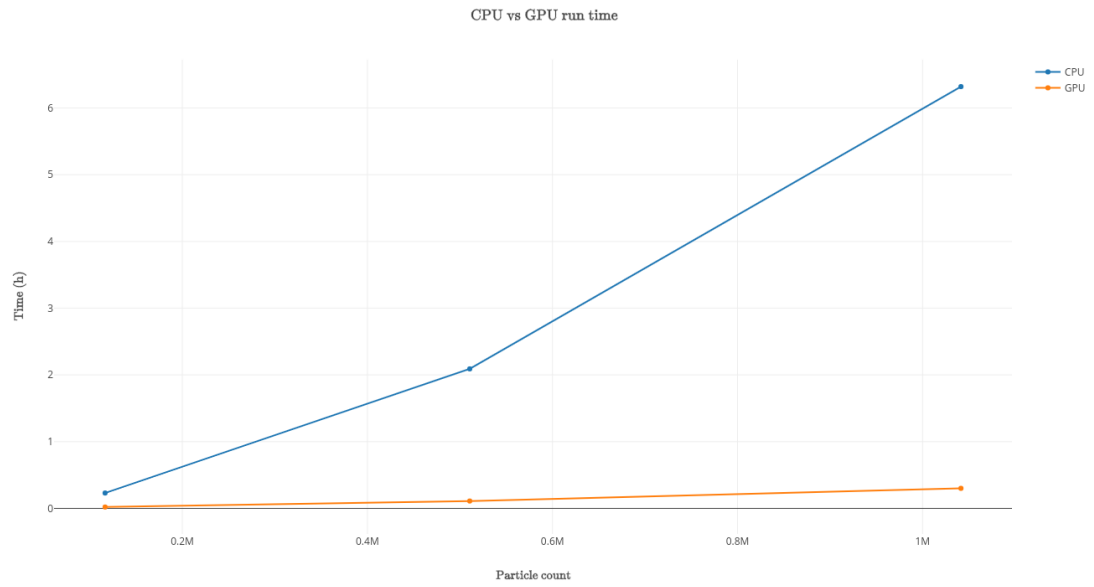


Figure 4.10: CPU and GPU comparisons.

As can be seen on the following tables, there was no significant change in memory usage or steps generated between the two turbulence methods. But using the RANS method the simulations were up to 10% faster. Taking into consideration that many simulations in the literature use more than 1 million particles, those 10% can become a great difference in simulation time with the increase of particle count.

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v1 (no model)	261218	22.42 MB	41.22 MB	933629	2.5 h
Baffle v1 (no model)	261218	22.42 MB	41.22 MB	937264	2.4 h
No baffle v1 (RANS)	261218	22.42 MB	53.17 MB	941603	2.6 h
Baffle v1 (RANS)	261218	22.51 MB	53.18 MB	947839	2.6 h
No baffle v1 (SPS)	261218	22.42 MB	53.17 MB	938048	2.6 h
Baffle v1 (SPS)	261218	22.42 MB	53.18 MB	942151	2.6 h

Table 4.1: Simulation data for $H = 0.6m$, $T = 1.3s$ and $dp = 0.002m$

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v1 (no model)	65812	5.65 MB	10.40 MB	465276	0.47 h
Baffle v1 (no model)	65812	5.65 MB	10.40 MB	466705	0.48 h
No baffle v1 (RANS)	65812	5.65 MB	13.41 MB	467651	0.5 h
Baffle v1 (RANS)	65812	5.65 MB	13.41 MB	469251	0.5 h
No baffle v1 (SPS)	65812	5.65 MB	13.41 MB	467899	0.51 h
Baffle v1 (SPS)	65812	5.65 MB	13.41 MB	469286	0.5 h

Table 4.2: Simulation data for $H = 0.6m$, $T = 1.3s$ and $dp = 0.004m$

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v1 (no model)	16632	1.43 MB	2.64 MB	231106	0.19 h
Baffle v1 (no model)	16632	1.43 MB	2.64 M	231675	0.19 h
No baffle v1 (RANS)	16632	1.43 MB	3.40 MB	232265	0.18 h
Baffle v1 (RANS)	16632	1.43 MB	3.40 MB	232692	0.18 h
No baffle v1 (SPS)	16632	1.43 MB	3.40 MB	232414	0.2 h
Baffle v1 (SPS)	16632	1.43 MB	3.40 MB	232690	0.19 h

Table 4.3: Simulation data for $H = 0.6m$, $T = 1.3s$ and $dp = 0.008m$

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v2 (no model)	218018	18.8 MB	35.43 MB	853644	1.85 h
Baffle v2 (no model)	218018	18.71 MB	34.41 MB	857010	1.99 h
No baffle v2 (RANS)	218018	18.71 MB	44.38 MB	858814	2.09 h
Baffle v2 (RANS)	218018	18.71 MB	44.39 MB	865241	2.09 h
No baffle v2 (SPS)	218018	18.71 MB	44.38 MB	858803	2.21 h
Baffle v2 (SPS)	218018	18.71 MB	44.39 MB	865942	2.1 h

Table 4.4: Simulation data for $H = 0.5m$, $T = 1.875s$ and $dp = 0.002m$

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v2 (no model)	55012	4.72 MB	8.70 MB	425096	0.55 h
Baffle v2 (no model)	55012	4.72 MB	8.70 MB	426675	0.54 h
No baffle v2 (RANS)	55012	4.72 MB	11.22 MB	428054	0.43 h
Baffle v2 (RANS)	55012	4.72 MB	11.22 MB	430506	0.43 h
No baffle v2 (SPS)	55012	4.72 MB	11.22 MB	427842	0.43 h
Baffle v2 (SPS)	55012	4.72 MB	11.22 MB	430510	0.43 h

Table 4.5: Simulation data for $H = 0.5m$, $T = 1.875s$ and $dp = 0.004m$

Case	Number of particles	Memory (CPU)	Memory (GPU)	Number of steps	Simulation time
No baffle v2 (no model)	14052	1.21 MB	2.23 MB	211667	0.2 h
Baffle v2 (no model)	14052	1.21 MB	2.23 MB	212376	0.2 h
No baffle v2 (RANS)	14052	1.29 MB	2.87 MB	213082	0.2h
Baffle v2 (RANS)	14052	1.21 MB	2.87 MB	214049	0.23h
No baffle v2 (SPS)	14052	1.21 MB	2.87 MB	213087	0.22 h
Baffle v2 (SPS)	14052	1.21 MB	2.87 MB	214184	0.22 h

Table 4.6: Simulation data for $H = 0.5m$, $T = 1.875s$ and $dp = 0.008m$

Chapter 5

Conclusion

Fluid simulation is a great tool for many areas of study. The applications can range from pure entertainment to more critical usages such as naval engineering.

In this work we chose to study the sloshing phenomenon, using different methods for turbulence modeling and were able to validate the models used.

Depending on the purpose of the application, the use of turbulence models may not bring greater insights. Many applications don't need realistic representation of the inner part of the fluid and choosing to use turbulence models might not lead to better results. But, when there is need for better vortex and inner fluid representation, the advantages of using turbulence models quickly becomes clear.

5.1 Future works

As next steps a few options are listed below, not in any particular order:

- Extend the simulations to 3D.
- Multi-phase simulations to study the damping effects of air or other gases on the liquid phase. In the real world the tank is usually filled with air, adding an air phase to the simulation may improve the results[38].
- Scenarios with floating objects and more violent flows. There already exists in the literature experiments for sloshing cases with floating baffles and other objects, aiming for sloshing reduction[39][40].

References

[1] J.-C. Piao, J.-M. Lu, C.-P. Hong, and S.-D. Kim, "Lightweight particle-based real-time fluid simulation for mobile environment," *Simulation Modelling Practice and Theory*, vol. 77, pp. 32–48, 2017, ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2017.05.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X17300850.

[2] L. d. S. R. Neto, F. D. Guimaraes, A. L. Apolinario Jr., and V. M. Mello, "Real-time screen space rendering of cartoon water," *SBGames*, vol. XII, 2013.

[3] C. Wojtan, M. Müller-Fischer, and T. Brochu, "Liquid simulation with mesh-based surface tracking," in *ACM SIGGRAPH 2011 Courses*, ser. SIGGRAPH '11, Vancouver, British Columbia, Canada: ACM, 2011, 8:1–8:84, ISBN: 978-1-4503-0967-7. DOI: 10.1145/2037636.2037644. [Online]. Available: http://doi.acm.org/10.1145/2037636.2037644.

[4] C. Braley, V. Tech, and A. Sandu, "Fluid simulation for computer graphics: A tutorial in grid based and particle based methods," 2009.

[5] M. B. Liu and G. R. Liu, "Smoothed particle hydrodynamics (sph): An overview and recent developments," *Archives of Computational Methods in Engineering*, vol. 17, no. 1, pp. 25–76, Mar. 2010, ISSN: 1886-1784. DOI: 10.1007/s11831-010-9040-7. [Online]. Available: https://doi.org/10.1007/s11831-010-9040-7.

[6] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '03, San Diego, California: Eurographics Association, 2003, pp. 154–159, ISBN: 1-58113-659-5. [Online]. Available: http://dl.acm.org/citation.cfm?id=846276.846298.

[7] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society*, vol. 181, pp. 375–389, Nov. 1977. DOI: 10.1093/mnras/181.3.375.

[8] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH Fluids in Computer Graphics," in *Eurographics 2014 - State of the Art Reports*, S. Lefebvre and M. Spagnuolo, Eds., The Eurographics Association, 2014. DOI: 10.2312/egst.20141034.

[9] F. Reichl, M. Treib, and R. Westermann, "Visualization of big SPH simulations via compressed octree grids," pp. 71–78. DOI: 10.1109/BigData.2013.6691717. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6691717.

[10] D. Violeau and B. D. Rogers, "Smoothed particle hydrodynamics (sph) for free-surface flows: Past, present and future," *Journal of Hydraulic Research*, vol. 54, no. 1, pp. 1–26, 2016. DOI: 10.1080/00221686.2015.1119209. eprint: https://doi.org/10.1080/00221686.2015.1119209. [Online]. Available: https://doi.org/10.1080/00221686.2015.1119209.

[11] F. Durier and C. Dalla Vecchia, "Implementation of feedback in smoothed particle hydrodynamics: towards concordance of methods," *Monthly Notices of the Royal Astronomical Society*, vol. 419, pp. 465–478, Jan. 2012. DOI: 10.1111/j.1365-2966.2011.19712.x. arXiv: 1105.3729 [astro-ph.CO].

[12] L. Tavares da Silva and G. A. Giraldi, "Variational Time Integration Approach for Smoothed Particle Hydrodynamics Simulation of Fluids," *ArXiv e-prints*, Dec. 2015. arXiv: 1512.04444 [physics.comp-ph].

[13] C. Altomare, A. Crespo, J. Dominguez, M. Gomez-Gesteira, T. Suzuki, and T. Verwaest, "Applicability of smoothed particle hydrodynamics for estimation of sea wave impact on coastal structures," eng, *COASTAL ENGINEERING*, vol. 96, pp. 1–12, 2015, ISSN: 0378-3839. [Online]. Available: http://dx.doi.org/10.1016/j.coastaleng.2014.11.001.

[14] A. M. Xenakis, S. J. Lind, P. K. Stansby, and B. D. Rogers, "Landslides and tsunamis predicted by incompressible smoothed particle hydrodynamics (sph) with application to the 1958 lituya bay event and idealized experiment," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2199, 2017, ISSN: 1364-5021. DOI: 10.1098/rspa.2016.0674. eprint: http://rspa.royalsocietypublishing.org/content/473/2199/20160674.full.pdf. [Online]. Available: http://rspa.royalsocietypublishing.org/content/473/2199/20160674.

[15] X. Dong, G. Liu, Z. Li, and W. Zeng, "A smoothed particle hydrodynamics (sph) model for simulating surface erosion by impacts of foreign particles," English, *Tribology International*, vol. 95, no. Complete, pp. 267–278, 2016. DOI: 10.1016/j.triboint.2015.11.038.

[16] J. R. Shao, H. Q. Li, G. R. Liu, and M. B. Liu, "An improved sph method for modeling liquid sloshing dynamics," *Comput. Struct.*, vol. 100-101, pp. 18–26, Jun. 2012, ISSN: 0045-7949. DOI: 10.1016/j.compstruc.2012.02.005. [Online]. Available: http://dx.doi.org/10.1016/j.compstruc.2012.02.005.

[17] N. N. Moiseyev and V. V. Rumyantsev, *Dynamic stability of bodies containing fluid*. Springer, Berlin, Heidelberg, 1968. DOI: https://doi.org/10.1007/978-3-642-86452-0.

[18] G. S. Brar and S. Singh, "An experimental and cfd analysis of sloshing in a tanker," *Procedia Technology*, vol. 14, pp. 490–496, 2014, 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014, ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2014.08.062. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221201731400098X.

[19] J. Grant, M. Prakash, S. E. Semercigil, and Ö. F. Turan, "Sloshing and energy dissipation in an egg: Sph simulations and experiments," *Journal of Fluids and Structures*, vol. 54, pp. 74–87, 2015, ISSN: 0889-9746. DOI: https://doi.org/10.1016/j.jfluidstructs.2014.10.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0889974614002229.

[20] J. Cappello, A. Sauret, F. Boulogne, E. Dressaire, and H. A. Stone, "Damping of liquid sloshing by foams: From everyday observations to liquid transport," *Journal of Visualization*, vol. 18, no. 2, pp. 269–271, May 2015, ISSN: 1875-8975. DOI: 10.1007/s12650-014-0250-1. [Online]. Available: https://doi.org/10.1007/s12650-014-0250-1.

[21] F. Cheli, V. D'Alessandro, A. Premoli, and E. Sabbioni, "Simulation of sloshing in tank trucks," vol. 20, pp. 1–16, Jan. 2013.

[22] A. Crespo, J. Domínguez, B. 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*, vol. 187, pp. 204–216, 2015, ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2014.10.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465514003397.

[23] J. Monaghan, "Simulating free surface flows with sph," *Journal of Computational Physics*, vol. 110, no. 2, pp. 399–406, 1994, ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1994.1034. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999184710345.

[24] J. J. Monaghan and J. C. Lattanzio, "A refined particle method for astrophysical problems," *ASTRONOMY AND ASTROPHYSICS*, vol. 149, pp. 135–143, Aug. 1985.

[25] H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 389–396, Dec. 1995, ISSN: 1572-9044. DOI: 10.1007/BF02123482. [Online]. Available: https://doi.org/10.1007/BF02123482.

[26] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual Review of Astronomy and Astrophysics*, vol. 30, no. 1, pp. 543–574, 1992. DOI: 10.1146/annurev.aa.30.090192.002551. eprint: https://doi.org/10.1146/annurev.aa.30.090192.002551. [Online]. Available: https://doi.org/10.1146/annurev.aa.30.090192.002551.

[27] E. Y. Lo and S. Shao, "Simulation of near-shore solitary wave mechanics by an incompressible sph method," *Applied Ocean Research*, vol. 24, no. 5, pp. 275–286, 2002, ISSN: 0141-1187. DOI: https://doi.org/10.1016/S0141-1187(03)00002-6. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141118703000026.

[28] H. Gotoh, S. Shao, and T. Memita, "Sph-les model for numerical investigation of wave interaction with partially immersed breakwater," *Coastal Engineering Journal*, vol. 46, no. 1, pp. 39–63, 2004. DOI: 10.1142/S0578563404000872. eprint: https://doi.org/10.1142/S0578563404000872. [Online]. Available: https://doi.org/10.1142/S0578563404000872.

[29] R. Dalrymple and B. Rogers, "Numerical modeling of water waves with the sph method," *Coastal Engineering*, vol. 53, no. 2, pp. 141–147, 2006, Coastal Hydrodynamics and Morphodynamics, ISSN: 0378-3839. DOI: https://doi.org/10.1016/j.coastaleng.2005.10.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378383905001304.

[30] L. Blin, A. Hadjadj, and L. Vervisch, "Large eddy simulation of turbulent flows in reversing systems*," *Journal of Turbulence*, vol. 4, 1, p. 1, Jan. 2003. DOI: 10.1088/1468-5248/4/1/001.

[31] L. Verlet, "Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules," *Phys. Rev.*, vol. 159, pp. 98–103, 1 Jul. 1967. DOI: 10.1103/PhysRev.159.98. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.159.98.

[32] B. J. Leimkuhler, S. Reich, and R. D. Skeel, "Integration methods for molecular dynamics," in *Mathematical Approaches to Biomolecular Structure and Dynamics*, J. P. Mesirov, K. Schulten, and D. W. Sumners, Eds. New York, NY: Springer New York, 1996, pp. 161–185, ISBN: 978-1-4612-4066-2. DOI:

10.1007/978-1-4612-4066-2_10. [Online]. Available: https://doi.org/10.1007/978-1-4612-4066-2_10.

[33] A. C. Crespo, J. M. Dominguez, 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*, vol. 6, no. 6, pp. 1–13, Jun. 2011. DOI: 10.1371/journal.pone.0020685. [Online]. Available: https://doi.org/10.1371/journal.pone.0020685.

[34] R. B. Canelas, A. J. Crespo, J. M. Domínguez, R. M. Ferreira, and M. Gómez-Gesteira, "Sph–dcdem model for arbitrary geometries in free surface solid–fluid flows," *Computer Physics Communications*, vol. 202, pp. 131–140, 2016, ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2016.01.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465516000254.

[35] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008, ISSN: 1542-7730. DOI: 10.1145/1365490.1365500. [Online]. Available: http://doi.acm.org/10.1145/1365490.1365500.

[36] O. Faltinsen, O. F. Rognebakke, I. A. Lukovsky, and A. Timokha, "Multidimensional modal analysis of nonlinear sloshing in a rectangular tank with finite water depth," vol. 407, pp. 201–234, Mar. 2000.

[37] G. Z, "A brief survey of nonparametric statistics," vol. 5, pp. 429–453, Jan. 1976.

[38] A. Mokos, B. D. Rogers, P. K. Stansby, and J. M. Domínguez, "Multi-phase sph modelling of violent hydrodynamics on gpus," *Computer Physics Communications*, vol. 196, pp. 304–316, 2015, ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2015.06.020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465515002659.

[39] Y.-M. Yu, N. Ma, S.-M. Fan, and X.-C. Gu, "Experimental and numerical studies on sloshing in a membrane-type lng tank with two floating plates," *Ocean Engineering*, vol. 129, pp. 217–227, 2017, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2016.11.029. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801816305327.

[40] K. Pan, X. Li, Y. Fan, and N. Cui, "The design of the floating baffle for cassini tank and the analysis of restraining sloshing," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 232, no. 3, pp. 448–458, 2018. DOI: 10.1177/0954410016680379. eprint: https://doi.org/10.1177/0954410016680379. [Online]. Available: https://doi.org/10.1177/0954410016680379.

Appendix A

Algorithm 1 Python implementation

```
import pandas as pd
import numpy as np
import time as tm

start_time = tm.process_time()

desiredx = 0.05
k = 0.001
nbfiles = 1001
rhoRes = 0.01
surfaceRhoBase = 1000
auxRhoResIncreaseFactor = 0.001
auxKincreaseFactor = 0.001
initHeight = 0
surfacecount = 0

csv_time_start = tm.process_time()

for j in range(0, nbfiles):

 print(str(round(((j / nbfiles) * 100),2)) + '%')

 filenb = '{:04d}'.format(j)

 boundfile = 'partboundcsv_' + filenb + '.csv'
```

```
    fluidfile = 'Partfluidcsv_' + filenb + '.csv'

    formatedb = 'formboundcsv_' + filenb + '.csv'

    formatedf = 'formfluidcsv_' + filenb + '.csv'

    with open(boundfile, 'r') as bf, open(formatedb, 'w+') as
        fbf:

     bfl = bf.readlines()

     for l in bfl[3:]:
      fbf.write(l[:-2]+'\n')



    with open(fluidfile, 'r') as ff, open(formatedf, 'w+') as
        fff:

     ffl = ff.readlines()

     for l in ffl[3:]:
      fff.write(l[:-2]+'\n')

csv_time_end = tm.process_time()

plotter_time_start = tm.process_time()

timerHeaders = ['TimeStep [s]', 'Np', 'Nbound', 'Nfixed', '
    Nmoving', 'Nfloat', 'Nfluid']
fluidHeaders = ['Pos.x [m]', 'Pos.y [m]', 'Pos.z [m]', 'Idp
    ', 'Rhop [kg/m^3]']
boundHeaders = ['Pos.x [m]', 'Pos.y [m]', 'Pos.z [m]', 'Idp
    ']
maxZvalues = np.zeros(nbfiles)
times = np.zeros(nbfiles)
measurepos = np.zeros(nbfiles)

initialBound = pd.read_csv('formboundcsv_0000.csv',
```

```
    delimiter=';', header=0, names=boundHeaders)
boundID = (np.abs(initialBound['Pos.x [m]'].values -
    desiredx)).argmin()
realinitial = initialBound['Pos.x [m]'].iloc[boundID]

for j in range(0, nbfiles):
 print(str(round(((j / nbfiles) * 100),2)) + '%')
 filenb = '{:04d}'.format(j)

 boundfile = 'formboundcsv_' + filenb + '.csv'

 fluidfile = 'formfluidcsv_' + filenb + '.csv'

 timefile = 'partboundcsv_' + filenb + '.csv'

 bound = pd.read_csv(boundfile, delimiter=';', header=0,
    names=boundHeaders)

 fluid = pd.read_csv(fluidfile, delimiter=';', header=0,
    names=fluidHeaders)

 time = pd.read_csv(timefile, delimiter=';', header=0,
    names=timerHeaders, nrows=1)

 times[j] = time['TimeStep [s]'].iloc[0]

 timestepX = bound['Pos.x [m]'].iloc[boundID]
 measurepos[j] = timestepX

 indexes = []
 surfIdx = []

 # Filtering using rho
 aux = fluid['Rhop [kg/m^3]'].values
 auxRhoRes = rhoRes

 if (j ==0 ):
  while len(surfIdx) == 0:
   for i,x in enumerate(aux):
    if(abs(x - surfaceRhoBase) < auxRhoRes):
```

```
      surfIdx.append(i)
    auxRhoRes += auxRhoResIncreaseFactor
 else:
  while len(surfIdx) < surfacecount:
   for i,x in enumerate(aux):
    if(abs(x - surfaceRhoBase) < auxRhoRes):
     surfIdx.append(i)
   auxRhoRes += auxRhoResIncreaseFactor

 print('\nsurface indexes count: ')
 print(len(surfIdx))
 print()

 # Filtering along x axis
 auxK = k
 while len(indexes) == 0:
  for i,x in enumerate(surfIdx):
   if(abs(fluid['Pos.x [m]'].iloc[x] - timestepX) < auxK):
    indexes.append(x)
  auxK += auxKincreaseFactor

 zValues = np.zeros(len(indexes))
 for i,x in enumerate(indexes):
  zValues[i] = (fluid['Pos.z [m]'].iloc[x])

 maxZvalues[j] = np.average(zValues)
 if(j == 0):
  initHeight = maxZvalues[j]
  surfacecount = len(surfIdx)


maxZvaluesVariation = maxZvalues - initHeight
with open('heightsGraph.csv', 'w+') as graph:

  graph.write('x,z,bx\n')

  for i,x in enumerate(times):
   graph.write(str(x) + ',' + str(maxZvaluesVariation[i]) +
       ',' + str(measurepos[i] - realinitial) + '\n')
```

```
plotter_time_end = tm.process_time()

report = ('---plotter report---\n' +
    'total runtime..... ' + str(round(plotter_time_end -
      start_time)) + '\n' +
    'csv format time.... ' + str(round(csv_time_end -
      csv_time_start)) + '\n' +
    'plotter time....... ' + str(round(plotter_time_end -
      plotter_time_start)) + '\n')

with open('report.txt', 'w+') as rprt:
 rprt.write(report)
```

Appendix B

Example of definition file

```
<?xml version="1.0" encoding="UTF-8" ?>
<case>
    <casedef>
        <constantsdef>
            <lattice bound="1" fluid="1" />
            <gravity x="0" y="0" z="-9.81" comment="
                Gravitational acceleration" units_comment="m
                /s^2" />
            <rhop0 value="1000" comment="Reference density
                of the fluid" units_comment="kg/m^3" />
            <hswl value="0" auto="true" comment="Maximum
                still water level to calculate speedofsound
                using coefsound" units_comment="metres (m)"
                />
            <gamma value="7" comment="Polytropic constant
                for water used in the state equation" />
            <speedsystem value="0" auto="true" comment="
                Maximum system speed (by default the dam-
                break propagation is used)" />
            <coefsound value="20" comment="Coefficient to
                multiply speedsystem" />
            <speedsound value="0" auto="true" comment="
                Speed of sound to use in the simulation (by
                default speedofsound=coefsound*speedsystem)"
                 />
            <coefh value="0.91924" comment="Coefficient to
                calculate the smoothing length (h=coefh*sqrt
```

```
            (3*dp^2) in 3D)" />
        <cflnumber value="0.2" comment="Coefficient to
            multiply dt" />
    </constantsdef>
    <mkconfig boundcount="240" fluidcount="10" />
    <geometry>
        <definition dp="0.004" units_comment="metres (m
            )">
            <pointmin x="-1" y="0" z="-1" />
            <pointmax x="2" y="0" z="2" />
        </definition>
        <commands>
            <mainlist>
                <setshapemode>dp | bound</setshapemode>
                <setdrawmode mode="full" />
                <setmkbound mk="0" />
                <drawbox>
                    <boxfill>all</boxfill>
                    <point x="0" y="-0.1" z="0" />
                    <size x="1.73" y="0.2" z="1.15" />
                </drawbox>
                <setmkfluid mk="1" />
                <fillbox x="0.8" y="0" z="0.3">
                    <modefill>void</modefill>
                    <point x="0" y="-0.1" z="0" />
                    <size x="1.73" y="0.2" z="0.6" />
                </fillbox>
                <shapeout file="" />
<setmkbound mk="2" />
                <drawpoint x="-0.5" y="0" z="-0.12" />
                <drawpoint x="1.9" y="0" z="1.5" />
            </mainlist>
        </commands>
    </geometry>
<motion>
 <objreal ref="0">
  <begin mov="1" start="0"/>
  <mvpredef id="1" duration="10">
   <file name="C:\Users\dquei\Documents\DualSPHyshics\TG\
       Sloshing\movfile.dat" fields="2" fieldtime="0"
```

```
        fieldx="1" />
    </mvpredef>
  </objreal>
</motion>
    </casedef>
    <execution>
        <parameters>
            <parameter key="PosDouble" value="1" comment="
                Precision in particle interaction 0:Simple, 
                1:Double, 2:Uses and saves double (default
                =0)" />
            <parameter key="StepAlgorithm" value="2"
                comment="Step Algorithm 1:Verlet, 2
                :Symplectic (default=1)" />
            <parameter key="VerletSteps" value="40" comment
                ="Verlet only: Number of steps to apply 
                Euler timestepping (default=40)" />
            <parameter key="Kernel" value="2" comment="
                Interaction Kernel 1:Cubic Spline, 2
                :Wendland (default=2)" />
            <parameter key="ViscoTreatment" value="1"
                comment="Viscosity formulation 1:Artificial,
                2:Laminar+SPS (default=1)" />
            <parameter key="Visco" value="0.01" comment="
                Viscosity value" /> % Note alpha can depend
                on the resolution. A value of 0.01 is
                recommended for near irrotational flows.
            <parameter key="ViscoBoundFactor" value="0"
                comment="Multiply viscosity value with 
                boundary (default=1)" />
            <parameter key="DeltaSPH" value="0.1" comment="
                DeltaSPH value, 0.1 is the typical value, 
                with 0 disabled (default=0)" />
            <parameter key="#Shifting" value="0" comment="
                Shifting mode 0:None, 1:Ignore bound, 2
                :Ignore fixed, 3:Full (default=0)" />
            <parameter key="#ShiftCoef" value="-2" comment=
                "Coefficient for shifting computation (
                default=-2)" />
            <parameter key="#ShiftTFS" value="1.5" comment=
```

```
    "Threshold to detect free surface. Typically
    1.5 for 2D and 2.75 for 3D (default=0)" />
<parameter key="RigidAlgorithm" value="1"
    comment="Rigid Algorithm 1:SPH, 2:DEM (
    default=1)" />
<parameter key="FtPause" value="0.0" comment="
    Time to freeze the floatings at simulation
    start (warmup) (default=0)" units_comment="
    seconds" />
<parameter key="CoefDtMin" value="0.05" comment
    ="Coefficient to calculate minimum time step
    dtmin=coefdtmin*h/speedsound (default=0.05)
    " />
<parameter key="#DtIni" value="0.0001" comment=
    "Initial time step (default=h/speedsound)"
    units_comment="seconds" />
<parameter key="#DtMin" value="0.00001" comment
    ="Minimum time step (default=coefdtmin*h/
    speedsound)" units_comment="seconds" />
<parameter key="#DtFixed" value="DtFixed.dat"
    comment="Dt values are loaded from file (
    default=disabled)" />
<parameter key="DtAllParticles" value="0"
    comment="Velocity of particles used to
    calculate DT. 1:All, 0:Only fluid/floating (
    default=0)" />
<parameter key="TimeMax" value="10" comment="
    Time of simulation" units_comment="seconds"
    />
<parameter key="TimeOut" value="0.01" comment="
    Time out data" units_comment="seconds" />
<parameter key="IncZ" value="0" comment="
    Increase of Z+" units_comment="decimal" />
<parameter key="PartsOutMax" value="1" comment=
    "%/100 of fluid particles allowed to be
    excluded from domain (default=1)"
    units_comment="decimal" />
<parameter key="RhopOutMin" value="700" comment
    ="Minimum rhop valid (default=700)"
    units_comment="kg/m^3" />
```

```
            <parameter key="RhopOutMax" value="1300"
                comment="Maximum rhop valid (default=1300)"
                units_comment="kg/m^3" />
        </parameters>
    </execution>
</case>
```