NTNU
Norwegian University of Science and Technology
Department of Marine Technology

M.Sc.THESIS

Title:	**Delivered:**
Numerical solution of Navier Stokes by use of SPH	14.16.2010
	Availability: Open
Student: John Eirik Gisetstad	**Number of pages:**

Abstract:

Smoothed Particle Hydrodynamics is a computational fluid dynamics method specially suited for large deformations and free surfaces. An introduction to SPH and the theory is given together with basic understanding of how to utilize the method. A C++ program code has been implemented to test the method on three different test cases. These are the evolution of a circular drop with an initial velocity field, calculation of hydrostatic pressure and the case of a broken dam. SPH is found to be applicable for these cases and shows good potential for further improvement.

Keyword:

Smoothed Particle Hydrdynamics
Navier-Stokes

Advisor:

Håvard Holm

Address:
NTNU
Department of Marine Technology
N-7491 Trondheim

Location
Marinteknisk Senter
O. Nielsens vei 10

Tel. +47 73 595501
Fax +47 73 595697

Summary

Smoothed Particle Hydrodynamics is in many hydrodynamic problems a very suitable method, especially those with large deformations and free surfaces. Simulations of large impacts and multiphase flows may be a strong area as well. The interpolating functions are easy to implement and a SPH program does not require a huge amount of programming code to function. Though to achieve good results for more complex problems many of the simple SPH approximations should be modified. For example there have been constructed many smoothing functions and to choose there should be taken into account factors such as computational cost, numerical stability, approximation to the Dirac Delta function, differentiability, compact support and more. There are several approximations for the density such as the summation approach and continuity approach, the continuity density approach appears to be better than the summation approach due to its computational cost advantage and its applicability at free surfaces. The boundary conditions have been modelled several ways as for example boundary particles, virtual particles on the boundary and in the region boundary. Several approximations for velocity and energy have been used. Other improvement exists such as the XSPH variant which prevents particles from penetrating each other and makes particles more ordered. For simulating real fluids such as water, viscosity should be implemented. The SPH method might be a mature method, but it seems as if there still is quite much research left. It is hard to find a general common opinion or documentation of which approaches are the best. All this implies that when implementing a program based on SPH there are many factors to consider for achieving the best results and highest efficiency.

The Smoothed Particles Hydrodynamics method has been implemented to simulate three test cases. It has been shown that the evolution of the axis of a drop with an initial velocity can be approximated with only a few percent errors. The pressure at the center of the drop is however not as close to the analytical solution as desired, but shows some potential.

In addition the method's stability and ability to reproduce hydrostatic pressure is somewhat tested by simulating a volume of water placed in a container. The pressure at a given depth is measured as the fluid reaches equilibrium. Results show a pressure which are oscillating around a value equal to the analytical pressure. The surface is calm, the pressure oscillation decreases but does not vanish and the method shows no instability in this case.

A broken dam is investigated as the last scenario. The deformation of the fluid gives an impression to be realistic. The pressure results are not given as numbers, but can to some extend be seen in the visualization. The pressure distribution shows large differences in neighboring particles, thus seem a bit off. However it is noticed that the pressure rises at important areas such as at the impact against the opposite wall.

It is found that a relatively basic implementation of the method gives approved results in terms of deformations. The results in pressure are not as great, but show potential. This implies that by some further work the SPH method has great potential in simulating suitable scenarios.

Preface

This paper is a result of the study of the SPH method that has been done in the master thesis. The goal of this thesis was to acquire knowledge and understanding of the method as well as acquiring experience in programming a method such as this. This paper tries to give a documentation of the study that has been done of the SPH theory and some existing implementations and give a report of the implementation done and the results from that. The C++ code can be found on a DVD at the back of the paper. This code is not to be mistaken for a fully usable analysis program with GUI etc, but as a documentation of the tool utilized for this thesis. It has been fun to try programming such a method, even though I hardly managed to implement all the things I would have liked to (parallel processing, shader support etc.). I would use the opportunity to thank my advisor Haavard Holm for helping me to find an interesting topic and giving advice.

Trondheim, 14.june 2009

John Eirik Gisetstad

Contents

Table of figures

Introduction

Smoothed Particle Hydrodynamics is a computational method first used for simulating astrophysical phenomenon, but has turned out to be suitable for much more. The name Smoothed Particle Hydrodynamics is often shortened to SPH. This method is a mesh free Lagrangian method, no mesh is needed and the coordinates move with the particles. The method's approach for finding values such as velocity and density of a particle is basically to approximate them by interpolating over the neighbouring particles. To make decent results there are much more to it than that though.

There are several numerical methods for solving fluid flows. The Finite Difference Method (FDM) and the Finite Element Method (FEM) are probably two of the most used ones. Those two methods are grid based and surely they are great for many problems, but not so applicable in all aspects. Creating a grid is a very time consuming part of the work. It is very difficult to create programs which automatically can create good grids, so most of the times one needs to manually create or edit the grid. Often the calculations will break down because of poor grid, resulting in the need to modify the grid or in worst case construct the entire grid all over again. Large deformations are one of the largest disadvantages these grid based methods have. If the deformations become large the grid needs to be remade and the cost of time and money increases. Thus these methods are not very well suited for modelling problems with deformable boundaries, free surfaces and moving material surfaces.

The SPH method on the other hand is as mentioned a mesh free particle method. Mesh free particle methods are very interesting because of their potential to provide accurate and stable numerical solutions for many complex boundary conditions. This because they do not use a grid/mesh for connectivity between nodes, but they use distributed nodes, or better described as particles, which can move freely and may be tracked by their coordinates. The SPH method is very suitable for simulating large deformations, free surfaces and moving surfaces. For example it is suitable to simulate the creation of a breaking wave and the air-pocket which may occur. It is easy to track the particle values since each particle have their coordinates and velocity stored in each time step. This makes it for example very easy to visualize the particles' movement. The method has of course some disadvantages as well, such as applying boundary conditions and not being very suitable for simulating boundary layers.

This paper gives an overview of the basics of the SPH method. The main discussed areas are fundamental theory, governing equations, boundary treatment, parallel implementation, smoothing functions, smoothing length and nearest neighbour search. Based on some of the theory there have been made some simulations by use of the SPH method. The second part of the paper will try to describe what have been tested and what the results are.

The C++ code can be found on a DVD at the back of the paper. This code is not to be mistaken for a fully usable analysis program with GUI etc. but as a documentation of the tool utilized for this thesis. All code is written in one .cpp file, this is for simplicity and for not being bothered with the header-files in C++.

Theory

Fundamentals

In smoothed particle hydrodynamics the nodes' properties are estimated from their neighbouring nodes by using particle approximation equations. Nodes may be better described as fluid cells, regions of space or as particles. Only neighbouring particles within the smoothing length h contribute to a particle's estimated properties. The smoothed length will be discussed later in this paper. This chapter will describe how the particle approximation equations are derived.

If the domain were not yet discretized into particles, a field value could be found from the following integral (Liu, 2003)

$$\langle f(r)\rangle = \int_{\Omega} f(r')W(r-r',h)dr' \tag{1}$$

where Ω is the domain of the problem and $W(r-r',h)$ is the *smoothing function*. The origin of this equation is the integral representation of a function as follows

$$f(x) = \int_{\Omega} f(x')\delta(x-x')dx' \tag{2}$$

where δ is the *Dirac delta function*.

The smoothing function W is often called a *kernel function, kernel smoothing function* or just *kernel* as well. The smoothing function should satisfy the following conditions:

$$\int_{\Omega} W(r-r',h) = 1 \tag{3}$$

and

$$\lim_{h\to 0} W(r-r',h) = \delta(r-r') \tag{4}$$

Further discussion of the smoothing function is made later in the paper.

If the fluid is discretized into a number of sub volumes, more easily named particles, equation (1) may be approximated with

$$\langle f(r)\rangle = \sum_{j=1}^{N} f(r_j)W(r-r_j,h)(dV)_J \tag{5}$$

where the summation is over all other particles than the current. If we further use the following relation between volume, mass and density

$$dV_j = \frac{m_j}{\rho_j} \tag{6}$$

the equation will look as following

$$\langle f(r)\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho(r_j)} f(r_j) W(r - r_j, h) \tag{7}$$

By using that equation and substituting $f(r)$with the density, we get

$$\langle \rho(r)\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho(r_j)} \rho(r_j) W(r - r_j, h) = \sum_{j=1}^{N} m_j W_{ij} \tag{8}$$

Equation (8) is often referred to as the *summation density approach* (Liu, 2003).

Derivatives

To approximate the spatial derivative it is simply to substitute $f(r)$ with $\nabla f(r)$ in equation (1) which gives

$$\langle \nabla * f(r)\rangle = \int_{\Omega} \nabla * f(r') W(r - r', h) dr'. \tag{9}$$

By integrating by parts and assuming that the solution domain Ω extends far enough so that on its boundaries the function $f(r)$ or W itself vanishes, the surface term will be identical to zero (Pákozdi, 2005), and we get the following equation

$$\langle \nabla * f(r)\rangle = \int_{\Omega} f(r') \nabla W(r - r', h) dr' \tag{10}$$

By using particle approximation the derivative can be expressed as

$$\langle \nabla f(r)\rangle = \sum_{j=1}^{N} \left(\frac{m_j}{\rho_j}\right) f(r_j) \nabla W(r - r_j, h) \tag{11}$$

The smoothing length

To reduce the computational cost, a smoothing length h is introduced. The smoothing length decides which particles are interacting neighbours and which are not. If two particles are at a greater distance from each other than the smoothing length, they will not take part in each other's calculations. To achieve best results there must be a sufficient amount of neighbours within the smoothing length of one particle. An excessive smoothing length will make the computational cost unnecessary large though. This problem could be solved by using variable smoothing length. A variable smoothing length may cause problems in momentum conservation since it is required that the same smoothing length is used when the force on particle j from particle i is calculated as well as the force on particle i from particle j. Some of the literature regarding SPH use $h * \alpha$ as the size of the area where particles influence each other, hence the smoothing length may not be as simple as just h in every case. Most of the examples in this paper use only h to describe the size of the influence domain and thus α equal to one. A simple example of how to implement a variable smoothing length is to update one particles smoothing length with the following expression

$$h = h_0 \left(\frac{\rho_0}{\rho}\right)^{\frac{1}{d}} \tag{12}$$

where h_0 is the original smoothing length, $\rho_0$the initial density and d is number of dimensions. This function makes use of the relation that a particle with large density equals many close neighbours and vice versa. To cope with different smoothing length on a pair of neighbouring particles, the smoothing length may be expressed by taking the average, minimum or maximum of the particles' smoothing length and use this on both (Liu, 2003).

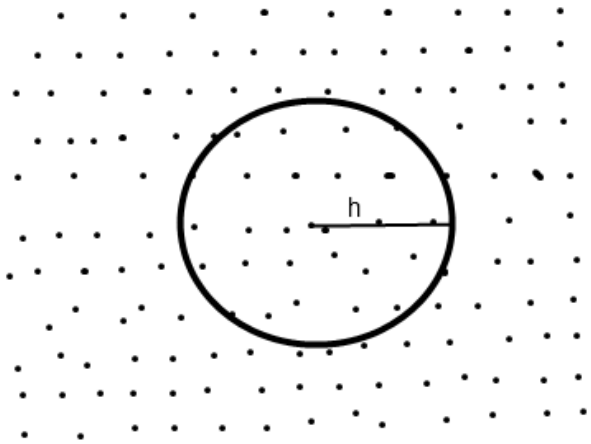


Figure 1 – A particles smoothing length and influence domain

In this influence area the "contribution" from each particle is weighted by the smoothing function W. This function may be physically described as the weighting factor which makes close particles have

large influence on each other while particles further apart have less. This will be further discussed in the next chapter.

The Smoothing function

The smoothing function, also called kernel, is an important part of the SPH method. By using interpolation one particle's properties is estimated from the neighbouring ones. The smoothing function gives a weight to each neighbouring particle which decides it's contribution to the value. The value from a distant particle is weighted low and a close particle is weighted high. This function takes part in estimating all results, thus it is important for accuracy and stability. Criteria for this will be discussed later. Computational cost of calculating the smoothing function is important as well considering that the function will be calculated for each of one particle's neighbours. That makes this function one of the most often computed in the simulation.

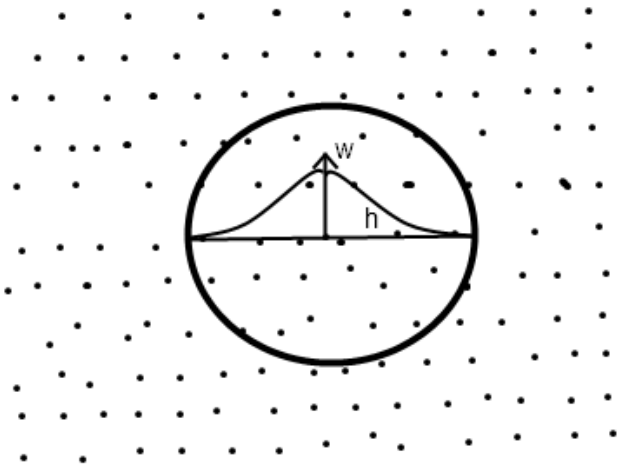


Figure 1 – Illustration of a smoothing function.

The smoothing function should meet the following criteria (Liu, 2003):

1. The smoothing function must be normalized (*Unity*) over its support domain (support domain is the area within the smoothed length).

$$\int_{\Omega} W(r - r', h)dr' = 1 \tag{13}$$

2. The smoothing function should be compactly supported.

$$W(r - r', h) = 0, \text{for } |r - r'| < h \tag{14}$$

3. For any point r' within the support domain for point r:

$$W(r - r') \geq 0 \tag{15}$$

4. The smoothing function value for a particle should be monotonically decreasing with the increase of the distance away from the particle.
5. The smoothing function should satisfy the Dirac delta function condition as the smoothing length approaches zero.

$$\lim_{h\to 0} W\,(r-r',h) = \delta(r-r') \tag{16}$$

6. The smoothing function should be an even function.
7. The smoothing function should be a sufficient smooth function.

A more detailed explanation of these criteria is given in *Smoothed Particle Hydrodynamics : A Meshfree Particle Method* by *Liu, G. R.* and *Liu, M. B* (Liu, 2003).

Several smoothing functions have been constructed for use in the SPH method. One used in (Pákozdi, 2005) are

$$W(r-r',h) = \frac{e^{\frac{-\|r-r'\|^2}{h^2}} - e^{-9}}{h^2\pi(1-10e^{-9})} \tag{17}$$

where $r-r'$ is the distance between two particles. Accordingly this function results in good stability properties and large code efficiency.

The governing equations

Three physical laws of conservation are essential in the SPH method for fluid flows. These are the Navier-Stokes equations and can be expressed as follows (Liu, 2003)

- The continuity equation

$$\frac{D\rho}{Dt} = -\rho\frac{\partial \mathbf{v}^{\boldsymbol{\beta}}}{\boldsymbol{\partial x^{\beta}}} \tag{18}$$

- Conservation of momentum (no external force in this case, then there should be extra term at the right hand side)

$$\frac{D\boldsymbol{v}^{\boldsymbol{\alpha}}}{Dt} = \frac{1}{\rho}\left(\frac{\partial \sigma^{\alpha\beta}}{\partial x^{\beta}}\right) \tag{19}$$

- The energy equation

$$\frac{De}{Dt} = \frac{\sigma^{\alpha\beta}}{\rho}\frac{\partial \boldsymbol{v}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} \tag{20}$$

Where $\sigma^{\alpha\beta} = -p\delta^{\alpha\beta} + \tau^{\alpha\beta}$.

By using $\tau^{\alpha\beta} = \mu\varepsilon^{\alpha\beta}$ the following energy equation can be derived (Liu, 2003)

$$\frac{De}{Dt} = -\frac{p}{\rho}\frac{\partial \boldsymbol{v}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} + \frac{\mu}{2\rho}\varepsilon^{\alpha\beta}\varepsilon^{\alpha\beta} \tag{21}$$

The continuity equation

The calculation of the density ρ may be the most important one in the SPH method. This because the density is the parameter used in the equation of state as well as in the dV_j term in the particle approximation equations. For a Lagrangian infinitesimal fluid cell[1] with volume of δV, the mass of the cell is

$$\delta m = \rho \delta V \tag{22}$$

where m is the mass and ρ is the density. By deriving this function and using that

$$\nabla \cdot \boldsymbol{v} = \frac{1}{\delta V}\left(\frac{D(\delta V)}{Dt}\right) \tag{23}$$

the following equation, mass conservation equation (18) , can be derived

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} \tag{24}$$

If particle approximation is applied the right hand side of this equation can be expressed as (Liu, 2003)

[1] infinitesimal fluid cell: Can be regarded as a very small clump of fluids associated with a very small control volume and a very small control surface surrounding the volume where the fluid properties are the same over the control volume (Liu, 2003).

$$\frac{D\rho}{dt} = \sum_{j=1}^{N} m_j v_{ij} \nabla W_{ij} \tag{25}$$

where $v_{ij} = v_i - v_j$ (velocity difference). Several density approximations have been derived, but equation (25) is probably the most frequently used (Liu, 2003). This is an alternative to the *summation density approach (8)* mentioned earlier and is called *continuity density approach.* These two approaches have their advantages and disadvantages though. The *summation density approach* conserves the total mass exactly in contrast to the *continuity density approach.* The summation approach requires more calculations and it may lead to spurious results due to edge effects. More calculation is needed because one pass is needed over the particles for calculating the mass, then another for calculating the rate of change of velocity. The *continuity density approach* on the other hand can calculate all rates of change in one pass. Averaging the density at the boundaries will include areas with no particles within the smoothing length (for example the free surface), thus wring density and spurious results. These and more advantages/disadvantages as well as possible modifications are discussed by Liu (Liu, 2003).

The momentum equation

The momentum equation (no gravity) is given in equation (18) as

$$\frac{D\boldsymbol{v}^{\boldsymbol{\alpha}}}{Dt} = \frac{1}{\rho}\left(\frac{\partial \sigma^{\alpha\beta}}{\partial x^{\beta}}\right) \tag{26}$$

By using different transformations these particle approximations can be derived in many ways, this is also the case with the momentum equation. One may stumble across a simplified version without viscosity (Schlatter, 1999)

$$\frac{D\boldsymbol{v}}{Dt} = -\frac{1}{\rho}\nabla P = -\sum_{j=1}^{N} m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2}\right) \nabla_i W_{ij} \tag{27}$$

Two very popular formulations are (Liu, 2003)

$$\frac{D\boldsymbol{v}^{\boldsymbol{\alpha}}}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta} + \sigma_j^{\alpha\beta}}{\rho_j\ \rho_i}\right)\left(\frac{\partial W_{ij}}{\partial x_i^{\beta}}\right) \tag{28}$$

and

$$\frac{D\boldsymbol{v}^{\boldsymbol{\alpha}}}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_\beta^2} \right) \left(\frac{\partial W_{ij}}{\partial x_i^\beta} \right) \tag{29}$$

It is worth to mention that the stress tensor includes both pressure force and viscous force.

The energy equation

These equations are all approximations, and the energy equation will look different depending on which approximations are used to derive it. According to Liu (Liu, 2003) the most popular expression for the pressure work (first term in the energy conservation equation) is

$$-\frac{P}{\rho}\left(\frac{\partial \mathbf{v}_\mathrm{i}^\beta}{\partial x_i^\beta}\right) = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \boldsymbol{v}_{ij}^\beta \left(\frac{\partial W_{ij}}{\partial x_i^\beta} \right) \tag{30}$$

By inserting this into the energy equation we get

$$\frac{De}{Dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \boldsymbol{v}_{ij}^\beta \left(\frac{\partial W_{ij}}{\partial x_i^\beta} \right) + \frac{\mu}{2\rho} \varepsilon_\mathrm{i}^{\alpha\beta} \varepsilon_\mathrm{i}^{\alpha\beta} \tag{31}$$

Artificial viscosity

Equation (27) may be used if one is not to include any form for viscosity. However to accurately simulate fluids, viscosity should be included. Several approaches have been proposed to include this. One of the most frequently seen is the one used by Monaghan (Monaghan, 1994)

$$\frac{D\boldsymbol{v_i}}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} + \prod_{ij} \right) \nabla_i W_{ij} + F_i \tag{32}$$

where m_j is the mass of particle j, P and ρ is pressure and density of corresponding particle, F is an external force (gravity), and $\prod_{ij}$ is the artificial viscos term given as

$$\prod_{ij} = \begin{cases} \dfrac{-\alpha c_{ij}\mu_{ij} + \beta\mu_{ij}^2}{\rho_{ij}}, if\ \boldsymbol{v_{ij}} \cdot \boldsymbol{r_{ij}} < 0 \\ 0, \quad otherwise \end{cases} \tag{33}$$

where

$$\mu_{ij} = \frac{h\boldsymbol{v_{ij}} \cdot \boldsymbol{r_{ij}}}{\boldsymbol{r_{ij}} + \eta^2} \tag{34}$$

where h is the smoothing length, $\eta^2 = 0.01h^2$, $\alpha = 0.1$ and $\beta = 0$. However these constants are situational, those values were suggested in (Batchelor, 1973) and (Monaghan, 1994).

The equation of state

An equation of state is required to completely describe the behaviour of the fluid by SPH. Monaghan (Monaghan, 1994) uses an equation of state given by Batchelor (Batchelor, 1973) which describes sound waves accurately. This equation of state, modified to suit lower speeds of sound, has the form

$$P = B\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right) \tag{35}$$

where $\gamma = 7$ and the choice of B determines the speed of sound. B determines how large the pressure force is for change in density. The (-1) in equation (35) is to act as an external pressure P_0 pushing back on the surface. This equation will give a large change of pressure for a small change in density which is reasonable for a close to incompressible fluid such as water. But it is worth to mention that a small error in the density will give a large error in the resulting pressure.

An alternative equation of state is

$$P = c^2(\rho - \rho_0) \tag{36}$$

where c is the speed of sound. This equation is not as sensitive for error in density as the previous equation. The approximation of the sound of speed c is very important though. The determination of the speed of sound can be done in several ways (Schlatter, 1999). One by utilizing equation for the Mach number

$$M = \frac{v}{c} \tag{37}$$

where v is the highest velocity expected and M is the Mach number. The sound of speed may be used to describe the level of incompressibility of the fluid. An incompressible fluid will have a high speed of sound. For simulating water it is desirable to have minimal variation in density. However

increasing the speed of sound will increase the pressure force in the equation of state for a given density, thus the particles must be moved in smaller time steps to avoid instability. For the density to vary about 1%, the Mach number should be equal to $0{,}1$. This gives a speed of sound $\sim 10v$ as a rule of thumb. Another way to determine c is to investigate the terms in the momentum equation to come up with relations that the speed of sound is proportional to. The speed of sound should be comparable with the largest of

$$c^2 \sim \frac{V_0^2}{\delta}, \frac{\nu V_0}{L_0 \delta}, \frac{F L_0}{\delta} \tag{38}$$

where $\delta = \frac{\Delta\rho}{\rho_0}$, V_0 is a velocity scale and L_0 is a length scale. (Schlatter, 1999).

XSPH

The velocity of a particle may be found from the simple time derivation of the position

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i \tag{39}$$

But there is another approach, the XSPH approach, which adds an extra term. This looks as following

$$\frac{d\boldsymbol{r}_i}{dt} = \hat{\boldsymbol{v}}_i = \boldsymbol{v}_i + \epsilon \sum_{j=0}^{N} m_j \frac{(\boldsymbol{v_j} - \boldsymbol{v_i})}{\overline{\rho_{ij}}} W_{ij} \tag{40}$$

where $\overline{\rho_{ij}} = \frac{\rho_i + \rho_j}{2}$ and $0 \leq \epsilon \leq 1$.

The XSPH approach was created by Monaghan for free surface flows. The term smoothes the velocity of the particles by using the relative velocity to their neighbours. It is stated that this approach tends to keep the particles more in order and that it prevents the penetration of one fluid by another in high speed flows (Batchelor, 1973). Keep in mind that if this variant is applied, the velocity term used in other equations must include this modification.

Boundary conditions

A grid based Lagrangian method has the advantage that it may set the nodes at the border to satisfy the border conditions and thereby easily take the border conditions into account, such as the FEM method. Introducing border conditions in the SPH method is not that easy though. The particles are to move freely and have no constant location/place in the particle distribution. Thus the particles

closest the boundary will not be the same at every time. In addition the field value interpolation will be wrong when a particle is near a rigid wall. The smoothed length will reach outside the boundary, as shown in the figure below, and gives faulty results. The velocity outside the border would be zero and correct, but the other field values such as the density would be wrong. This makes it necessary to make some sort of measures for treating the rigid walls.

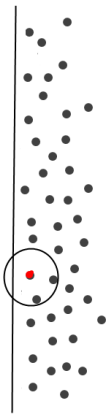


Figure 2 – Smoothed length of a particle extends outside the boundary

Several methods have been used for implementing these border conditions. In Monaghan's paper "Simulating free surface flows with SPH" (Monaghan, 1994) the boundary condition is treated by creating particles at the border which exert forces on the fluid particles. For a boundary and fluid particle separated with a distance r the force per unit mass $f(r)$has the Lennard-Jones form

$$f(r) = \frac{D\left(\left(\frac{r_0}{r}\right)^{p_1} - \left(\frac{r_1}{r}\right)^{p_2}\right)r}{r^2}, \text{ for } r > r_0$$
$$f(r) = 0 \quad , \text{otherwise} \qquad (41)$$

The constants p_1 and p_2 must satisfy the condition $p_1 > p_2$and for most of his simulations $p_1 = 4$ and $p_2 = 2$ were used. The length r_0 was taken to be the initial spacing between the particles. For simulations of dams, bores and weirs the coefficient D was used as $D = 5gH$ where H was the depth of the water. This approach prevents the particles from penetrating the boundary surface as it provokes a repulsing force when a particle comes close enough. To produce no-slip conditions the particles can be included in the calculation of the viscous term in the momentum equation (Monaghan, 1994).

Another approach is mentioned by Schlatter (Schlatter, 1999) where he refers to Morris who models the boundary using special smoothed particles as well but use such in a boundary region, not only as the boundary surface. The boundary region is filled uniformly with boundary particles and these particles contribute to the density such that the pressure decreases when a particle diverges from the boundary. The boundary particles do not have their position and velocity updated but they do evolve their density and hence their pressure. But if the boundary particles have zero velocity they cannot interact correctly with the fluid particles through the SPH equations. Morris has solved this by applying an artificial velocity to the boundary particles depending on the velocity of the interacting fluid particle. This approach has the following algorithm:

- For each free particle A
    - For each boundary particle B
        - Calculate the normal distance to the boundary from particle A
        - Calculate the tangent line in accordance with the normal of A
        - Calculate the normal distance to the tangent line from particle B
        - Extrapolate the velocity of free particle A across the tangent line, assuming zero velocity at tangent line , to particle B
    - End
- End

The relative velocity between a fluid particle and a neighbouring boundary particle can be expressed as

$$\boldsymbol{v}_{AB} = (1 + \left(\frac{d_B}{d_A}\right))\boldsymbol{v}_A = \beta \boldsymbol{v}_A \tag{42}$$

where d_A and d_B are the normal distance from the particle to the tangent line of the boundary. This relative velocity may grow towards infinity if the distance to the border goes towards zero. Thus a min-function is used

$$\beta = \min(\beta_{\max} \ , 1 + \frac{d_B}{d_A}) \tag{43}$$

where it is stated that a good value of β is 1.5.

Liu et al. (Liu, 2003) suggest using two types of particles to model the boundary. The first type is particles located at the boundary line, similar to what Monaghan used. The second type is virtual particles that fill the boundary region. The virtual particles of type two are constructed by that if a real particle is closer to the border than the distance αh. Then a symmetrically particle is constructed on the opposite side of the boundary.

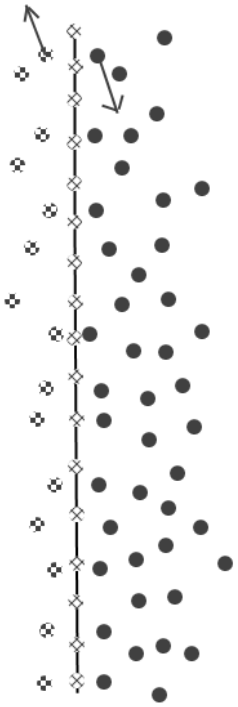


Figure 3 – Real particles close to a boundary modelled by two types of virtual particles

These virtual particles have the same density and pressure as the corresponding real particles, but have the opposite direction of the velocity. With the use of only particles of the second type, it may occur that particles penetrate the boundary. To prevent this from happening, the first type of particles is used in addition. Liu (Liu, 2003) states that this method with two types of virtual particles is very stable and effective.

Nearest neighbour search

SPH demands high resolution if the results are going to be good. Better resolution equals more particles and therefore more calculations. Thus it is important to make fast software code. One part of the SPH code that is especially time consuming is the nearest neighbouring particle search. This search involves finding the particles which are within one particles influence domain, in which the size of the domain is dependent on the smoothing length. This is one disadvantage the SPH method has compared to grid-based methods where the neighbours always are known. Nearest neighbour searching is a problem in many methods, not only in SPH, and several searching algorithms have been developed. Some of them will be discussed here.

The easiest way to do this is by traversing all the particles, calculate their distance from the present particle. If the distance is greater than the smoothing length (as long as the size of the influence domain is equal to the smoothing length) the particle will not have influence on the present particle. This would have to be done for all N particles for each particle. This implies that the computational time with this method would be $O(N^2)$. This has to be done in each time step and the computational cost for large problems becomes intolerable high.

One method for improving the cost of nearest neighbour search is to divide the problem domain into cells. The cells are used for book keeping of the particles. Each cell has a list of the particles that it contains. These list must be updated when a particles moves out from or into a cell. What cell size is favourable then? Using cell width and height that equal the smoothed length seems as the best idea. This way will a particle's neighbours always be in the same cell or in one of the neighbouring cells. As illustrated below this makes it necessary to search only nine cells in two dimensions.

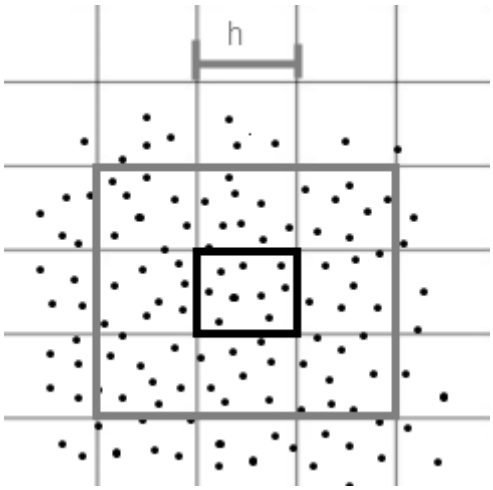


Figure 4 – Required searched cells for particles in the middle cell

In one and three dimensions the required amount of searched cells will be 3 and 27. This would drastically decrease the nearest neighbour search. This because if one cell contains in average 30 particles it would be necessary to iterate over only $30 * 9 = 270$ particles, and not all the particles in the problem area, which could be millions. The book keeping of particles in each cell will have a negative effect on the computational time though. One issue to be aware of is that this method becomes less efficient when a variable smoothing length is used. Less efficient because the cell size still would be constant and therefore optimal only for those particles with a smoothing length equal to the cells width and height. If the cells are very small compared to the total problem domain this method will be of order $O(N)$.

Tree algorithms are a good alternative approach to a solution, especially if a variable smoothing length is used. An example of this is to divide the entire problem into areas, areas which are further divided until one area contains only one particle. By making a tree of this, with the leaves to be the particles, one could easily search the tree for neighbours by eliminating the nodes (areas) which do not overlap with the influence domain of the particle, and thus not search through all the particles under the eliminated nodes.

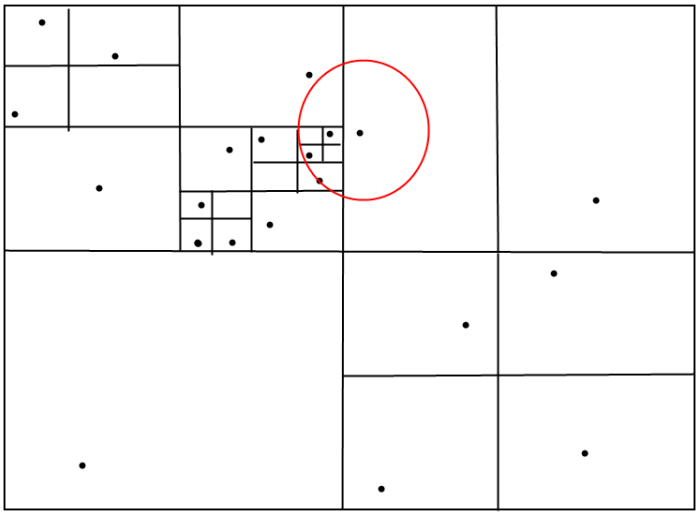


Figure 5 – Illustration of a divided domain for making a tree structure

Parallel processing

It might be useful to have a basic "picture" of how the algorithm of a typical SPH program looks like. Be aware that there may be several ways for such an implementation. The following one is close to the same as found in (Liu, 2003).

1. Initialization of geometry, boundaries, particle distribution, material values, time step (if required) etc.
2. Main SPH calculations which is done inside the time integration which standard methods such as Leapfrog, predictor-corrector and Runge-Kutta etc. can be applied for.
    a. Generation of boundary (virtual or ghost) particles.
    b. Nearest neighbouring particle searching. (Several methods to chose from, the fastest for a single CPU may not necessarily be the fastest for parallel processing.)
    c. Calculate the smoothing function (for the summation density approach) and its derivatives from the generated information of interaction particle pairs.
    d. Update density if the summation density approach is used.
    e. Calculate the artificial viscous force.
    f. Calculate the internal forces arising from the particle interactions. Note that the particle pressure is obtained from the density and energy through an equation of state.
    g. Calculate the change of momentum, energy and density while updating these properties. Checking the conservation of the energy and momentum could be done as well.
    h. Applying boundary conditions (how depends of course on how they are modelled).

3. Output. When the time step reaches a preferred time, a certain number of time steps is reached or other prescribed condition is reached the results should be saved. This would typically be velocity, position, pressure etc.

To achieve best possible results it is preferable to use a lot of particles. Thus the computational cost of simulation is high. Development of the performance of one single CPU-core has reached a temporary cap. So instead of improving each core the manufacturers of CPUs have begun producing CPUs with several cores. The reason for using of several CPUs could be to achieve lower computational times and to compute larger problems. Lower computational time since the work may be divided on multiple CPUs and larger problems since the data may be spread on each CPU's memory.

There are mainly two different types of parallel processing units. One type is where each core has its own memory (distributed memory), and the other one where the cores have shared memory. Some supercomputers have a mix though, a mix where a node (group of cores) share memory and can communicate with other nodes over a network. Often the code that is written for distributed memory systems will be compatible with shared memory systems but not the other way around. If the program is written in *Fortran*[2] or *C*[3]. Message Passing Interface (MPI) is a good API for parallel implementation.

How beneficial parallel implementation of a SPH program is depends on how much of the code that can be done in parallel. Code may be done in parallel if the present code line is not dependent on the result of the previous one. It is of course possible to make several cores do the same calculation, but that will not improve the computational time. A simple example of a problem that can be implemented for parallel processing may be to sum up a bunch of numbers. The numbers are divided evenly among all cores, each core has then a part of the sum, and these sums are then sent to one or all the cores. If the problem is to sum up the numbers in a series where number i is calculated with the value of $i-1$ there would be difficult to distribute the work. It is of course seldom that a program is that simple and very often only parts of programs can be done more efficient in parallel. The nearest neighbour search is as mentioned a time consuming part of the SPH method. This can be implemented with parallel processing by tasking each core to find the nearest neighbours of an equally amount of particles. If cells are used for book keeping each core could be assigned an area of cells, by this each core would only need the data from its own cells and the neighbouring cells. The data from the neighbouring cells must be achieved from other cores. This brings us to communication which is another factor that come into play in parallel performance. How the performance of putting more CPUs at work can be seen by studying the Speedup $S_{p,}$

$$S_p = \frac{T_1}{T_p} = \frac{1}{\frac{\alpha T_1}{P} + (1-\alpha)T_1 + T_{com}} \tag{44}$$

[2] Fortran: A programming language.
[3] C: A programming language.

where T_1 is the original computational time of the program, $T_p$is the computational time on P processors, α is the fraction of the code which is parallelizable and T_{comm} is the communicational cost between processors. Communicational cost is the amount of time used by the processors when they are communicating data. This communication cost is limited by the network bandwidth and the network latency. A describing comparison could be a phone call, bandwidth to be how many words you can say per second, and latency to be the time before the phone is picked up. This communication cost implies that there will be a point for each parallel program where it is not beneficial to add more CPUs to the calculation. It is also seen that the benefit from adding many CPUs to the calculation is low if the parallelization fraction α is little. Parallel implementation is beneficial for SPH programming, but the aspects mentioned should be taken into account.

Simulation

An implementation of the SPH method has been made to simulate:

- The evolution of a water drop
- Calm water with hydrostatic pressure
- Broken dam

The code is written in C++. C++ was chosen due to the high performance potential and good integration of the OpenGL API. Little experience in C++ during the education and thus a desire to learn a bit more was a factor as well.

The applied equations

The smoothing function

The smoothing function used in the simulation is (17). This function has a cutoff limit which will result in a value zero at a distance greater than *3h*. The constant terms in the function are computed only once to reduce the computational cost. Some investigation has been done to be positive of the function's applicability:

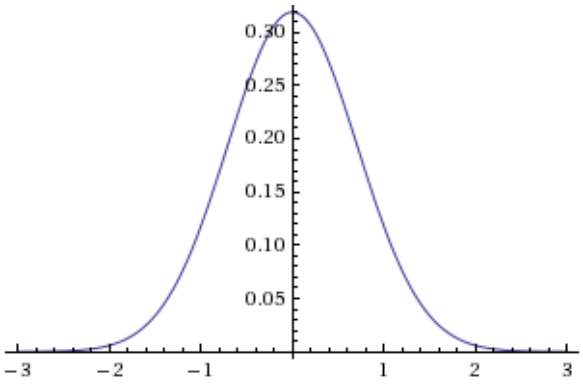


Figure 6 – Plot of the smoothing functionl used in the simulations

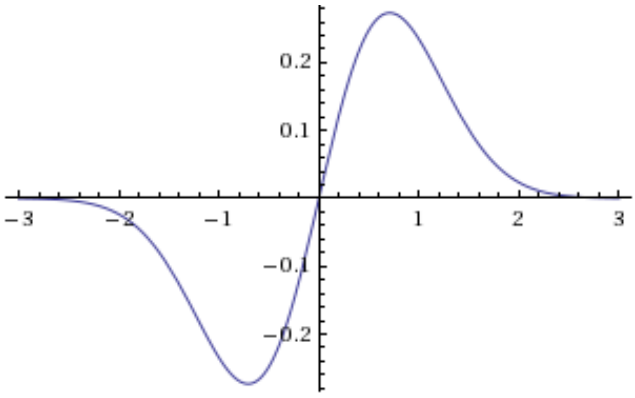


Figure 7 – The derivative of the kernel

The function is plotted in Figure 6 – Plot of the smoothing functionl used in the simulationswith $h = 1$ and $(r - r')$ as the horizontal axis. It is noticed that the value is decreasing with the increase of the distance and that the asymptotic value goes to zero for large input values. The cutoff limit was found to be at the value of three, just as promised.

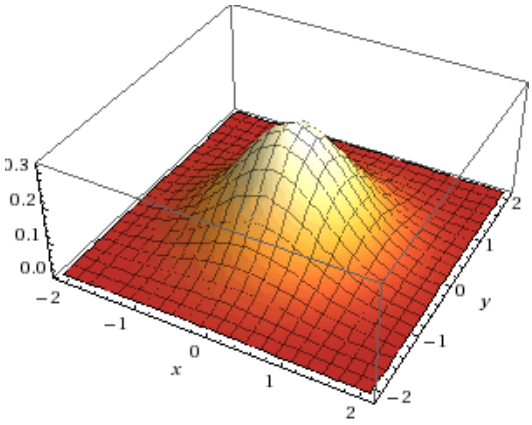


Figure 8 - The smoothing function's value plotted in he xy-plane

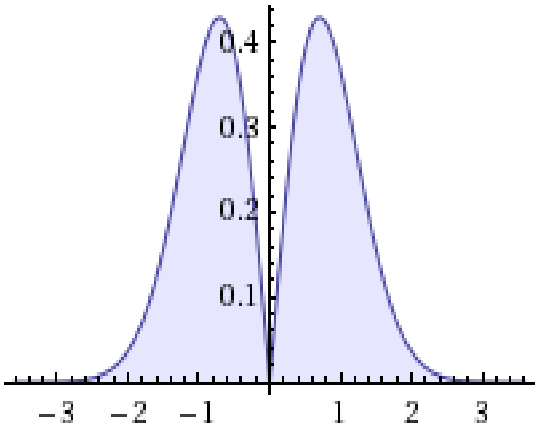


Figure 9– The integral of the smoothing function multiplied with πx

As mentioned the integral of the function over the domain should be equal to 1. By looking at Figure 6 the function may seem to give too small values, this is because the function is made for two dimensions and not one. As said in point 1 above the integral of the kernel should be equal to one in its domain. This is verified by multiplying the function with πx (half of the circumference of a circle) and integrated from -3 to 3 to attain the value in two dimensions.

$$\int_{-3}^{3} 1.00123562294\left(-0.00012341 + 2.71828182845^{-x^2}\right)|x|\,dx = 1. \qquad (45)$$

The constant terms in equation (17) are calculated forehand and inserted to give equation (45). The integral could be found by replacing the distance x with $\sqrt{(x^2+y^2)}$ and integrating in two dimensions as well. By inserting $(r-r') = s_{ab} = \sqrt{(x^2+y^2)}$ in (17) and derivate with respect to x and y, the derivatives is found as

$$\nabla_a W(s_{ab},h) = \begin{bmatrix} \frac{W(s_{ab},h)}{\partial x} \\ \frac{W(s_{ab},h)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{-2e^{\frac{-(s_{ab})^2}{h^2}}}{h^4\pi(1-10e^{-9})}(x_a - x_b) \\ \frac{-2e^{\frac{-(s_{ab})^2}{h^2}}}{h^4\pi(1-10e^{-9})}(y_a - y_b) \end{bmatrix} \quad (46)$$

Density approximation

Both the *summation density approach* (8) and the *continuity density approach (25)* were tested. The summation approach was applied first to verify the correctness of the particles initial positioning, mass per particle and the smoothing function. Particles were distributed evenly to form a drop, and given a mass equally the total mass of the circle divided on the number of particles. The correct densities were calculated as long as long as the particles were not at the boundary. This was expected due to the previously mentioned free surface flaw of the summation approach. Since free surface was to be simulated the *continuity density approach (25)* was used in the rest of the simulations.

The equation of state

To approximate the pressure for a particle, the following equation of state were applied

$$P = B[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1] \quad (47)$$

where γ was set to 7 and

$$B = \frac{\rho_0 c_s^2}{\gamma} \quad (48)$$

where c is the speed of sound and set to equal $\sim$10 times the highest expected velocity. However several values of B were tested as well as equation (36). Both equations were seemingly applicable for these simulations. The value of B related to the variation in density and time steps, just as predicted from the reasons mentioned earlier. As long as B was large enough, the results were good. Equation (48) worked well as a suggestion for B. Manually controlling the length of the time steps, the number of particles and testing different equations of state should be performed very carefully

to avoid unstable simulations. This instability is partly described in the chapter "The equation of state" earlier. It is mainly due to numerical flaws when the field properties are approximated by use of derivatives, the faster the forces change the smaller time steps must be applied.

Time stepping

When the particles' acceleration were found by use of equation (32) and the velocity adjusted with the XSPH approach (40) they were moved by using the relatively simple equation

$$s_{n+1} = s_n + \Delta t * v_{n+\frac{1}{2}} \tag{49}$$

where s is position,Δt is time step length and v is velocity.

However the lack of a better implemented time integration method with preferably a dynamic time step prediction was source for quite a few errors and frustration during the work.

Boundaries

Boundaries are needed to simulate the case of hydrostatic pressure and a breaking dam. In these cases the boundaries needed shall represent impenetrable walls. To achieve such boundaries particles were placed along the boundaries, particles that had a static position. The forces from these boundary particles were found by use of equation (41). In the visualization the boundary particles are given white color.

Visualization

A real-time visualization of the simulation is implemented. The OpenGL graphics API[4] is utilized. A framework from "nehe.gamedev.net" is used to give basic OpenGL functionality as starting ground. Each particle is represented by a GL_POINT and given color relative to its pressure. Most of the simulations are run with a small time step, typically 10^{-5}s, thus the graphics are set to update only at every 10th step.

However the best solution for visualization would be to send only the raw data to the GPU[5] and utilize its shaders. That would be a better solution because it makes the GPU calculate the graphics and not the CPU. In addition the GPU is more suited for calculating graphics, and more realistic visualization is possible. Programming by use of shaders would take quite some effort and the visualization in the written code is not the performance bottleneck, thus shaders were not utilized.

Evolution of a circular drop

A frequently used test case is the evolution of a circular drop. The drop is given velocities so that it is stretched in y-directions. The following initial velocities are given

[4] API: An **Application Programming Interface** (**API**) is an interface implemented by a software program which enables it to interact with other software. (www.Wikipedia.org)
[5] GPU: Graphics processing unit

$$v_x = -A_0 * x$$
$$v_y = A_0 * y \quad (50)$$

where $A_0$is a given velocity. No external forces exists but an initial pressure field is given (Jiannong Fang, 2008)

$$p_0 = \frac{1}{2}\rho_0 A_0^2(R^2 - (x^2 + y^2)) \quad (51)$$

where $\rho_0$is the density and R^2 is the radius of the drop. The drop shall remain elliptical during the simulation and the product of the semi-minor axis and the semi-major axis shall be constant. Incompressibility is the cause of this criterion.

In the simulation the radius is set to $1m$ and A_0 is set to $100m/s$ and 1270 particles are used. The evolution of the drop can be seen in the figures below.

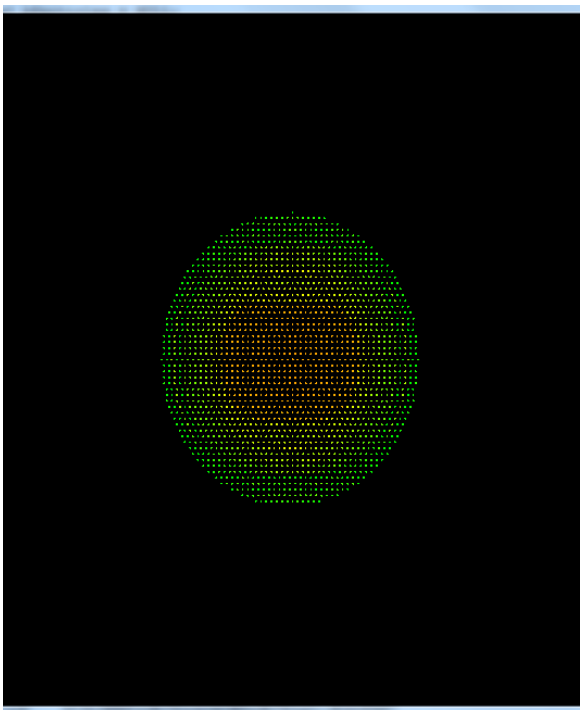


Figure 10 – Evolution of drop

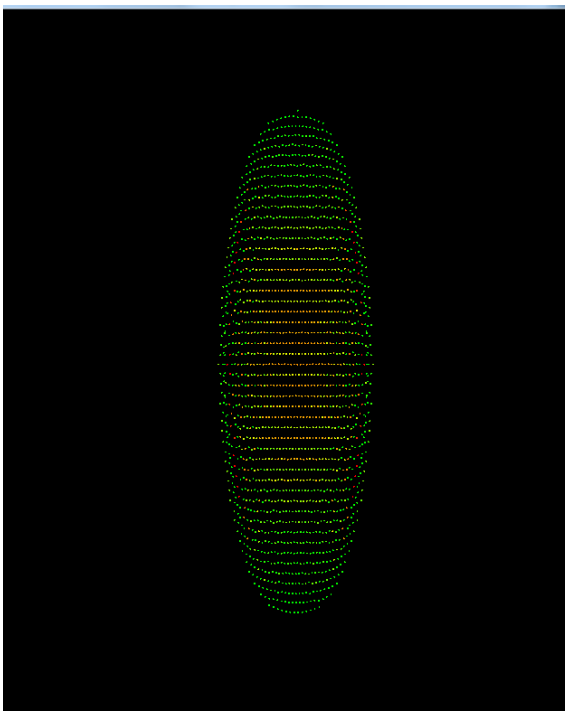


Figure 11 - Evolution of drop

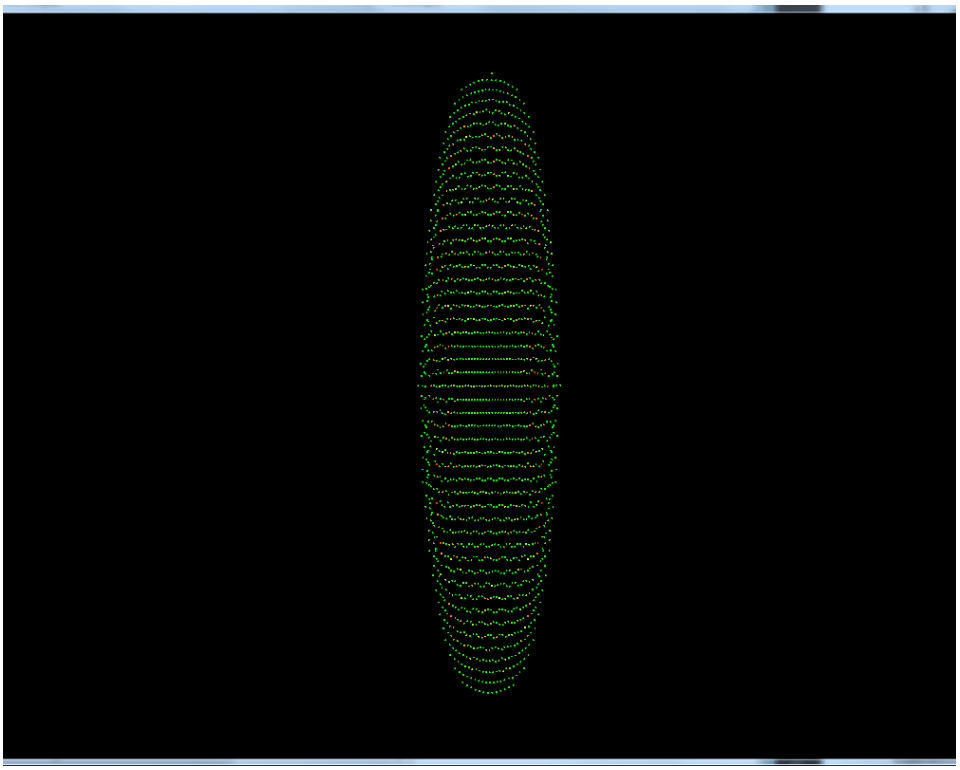


Figure 12 - Evolution of drop

At first glance from the visualization above, the drop seems to fulfil the criterion to remain elliptic. As well it is noticeable that the pressure is greatest at the middle, and decreases as the drop stretches, as expected. During the simulation results for the pressure, semi-minor axis and semi-major axis were stored. The length of the axis was found by selecting the coordinates of the particles that had the largest x-value and y-value. The pressure was taken as an average of the particles at the centre of the drop.

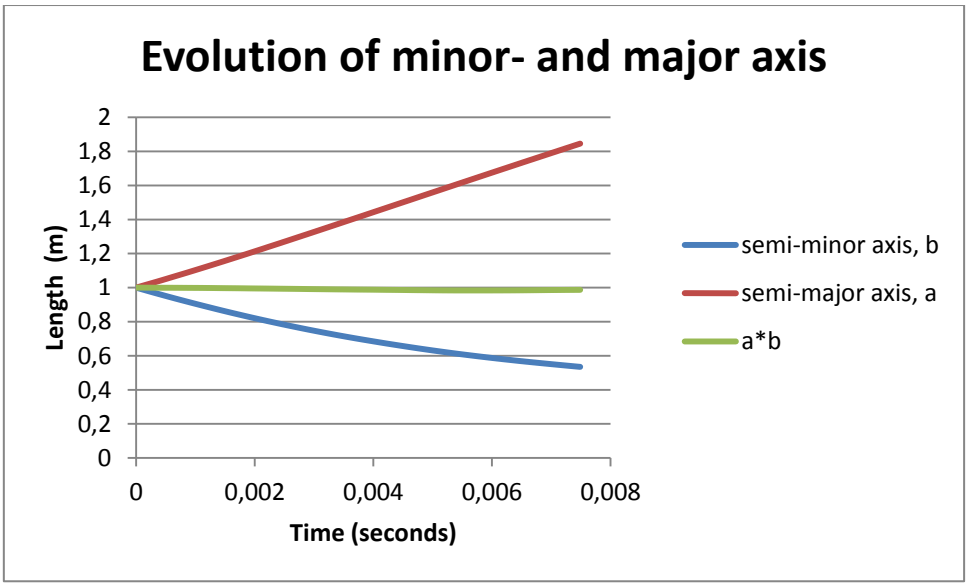


Figure 13 – Deformation results of the drop simulation

In the chart above the evolution of the axis can be seen. The criterion of a constant value of $a * b$ seems to be fulfilled. The length of the semi-major axis from analytical solution is (Monaghan, 1994)

Time(s)	Semi-Major Axis(theory)
0,0008	1,083
0,0038	1,44

where the same initial values have been used. The results agree well with $\sim 2\%$ error for the two controlling points. However as the drop evolves the results will most likely diverge more from the reason that the particles move further apart and less neighbours are attained due to the constant smoothing length.

The pressure graph on the other hand is not as nicely looking.

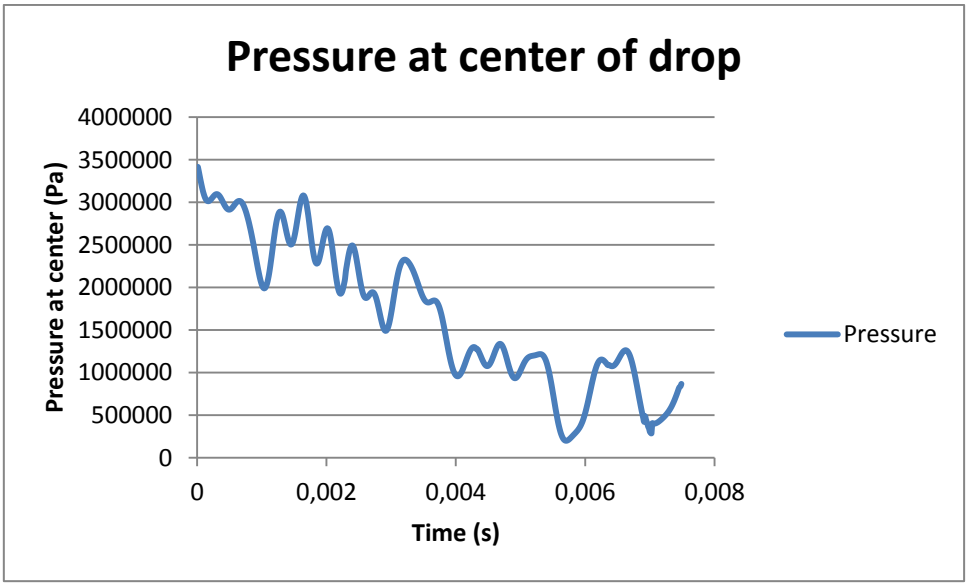
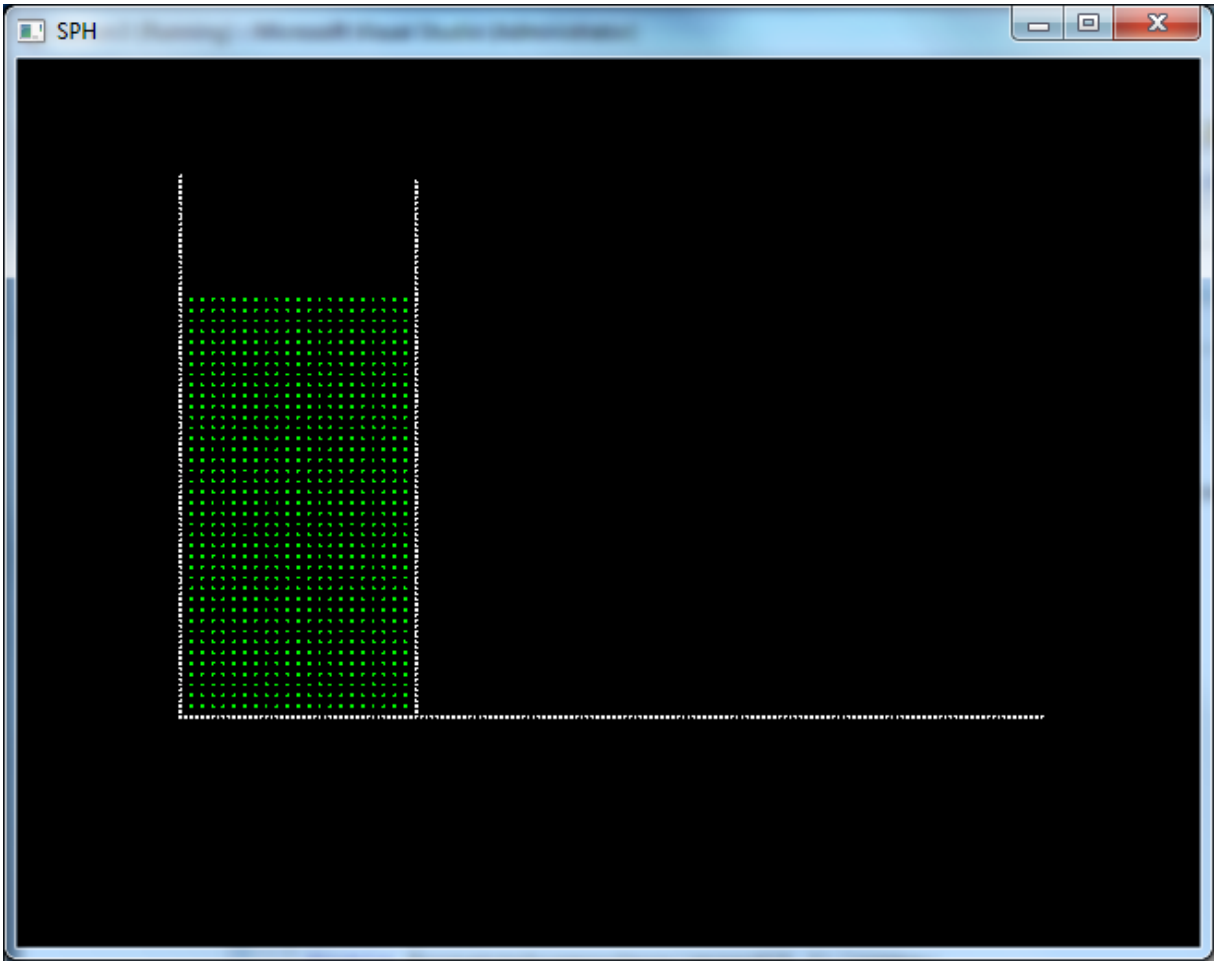


Figure 14 – Pressure at center of the drop

The initial pressure field should make the initial pressure at the center of the drop to be equal $\frac{1}{2}\rho A_0^2$. However the pressure above is taken as a mean of the particles within $abs(x) < 0{,}1$ and $abs(y) < 0{,}1$ and will thus be somewhat underestimated. The oscillating pressure results are typical for the SPH method due to that the fluid is approximated as slightly compressible. By comparing the results with the analytical results (Jiannong Fang, 2008) and taking the underestimating under consideration, it is seen that though the results oscillate a lot, it oscillates near the solution. However with such large oscillations, it can hardly give useful pressure results. The simulation was tested with a larger amount of particles as well, but did not improve the results.

Hydrostatic pressure

The nature of SPH should make it feasible to put a volume of water in a container and let the water reach equilibrium just as real water would. At equilibrium the pressure field should be equal to the hydrostatic pressure $p = \rho g h + p_0$. The surface pressure p_0 is set to zero for simplicity.



Figur 15 – Initial situation of the simulation

A container of height $5m$ and width $2m$ is filled with water to a height of $4m$. However the particles representing water is given no initial pressure, the same density $\rho = 1000 kg/m^3$ and are not calculated to initially be placed perfectly adjusted to the boundaries, but close though. The only external force is the gravity which is set to $g = 9{,}81 m/s^2$. The desired result is that the particles will move towards the borders and achieve an state of equilibrium. At equilibrium the particles' density shall give the hydrostatic pressure through the equation of state.

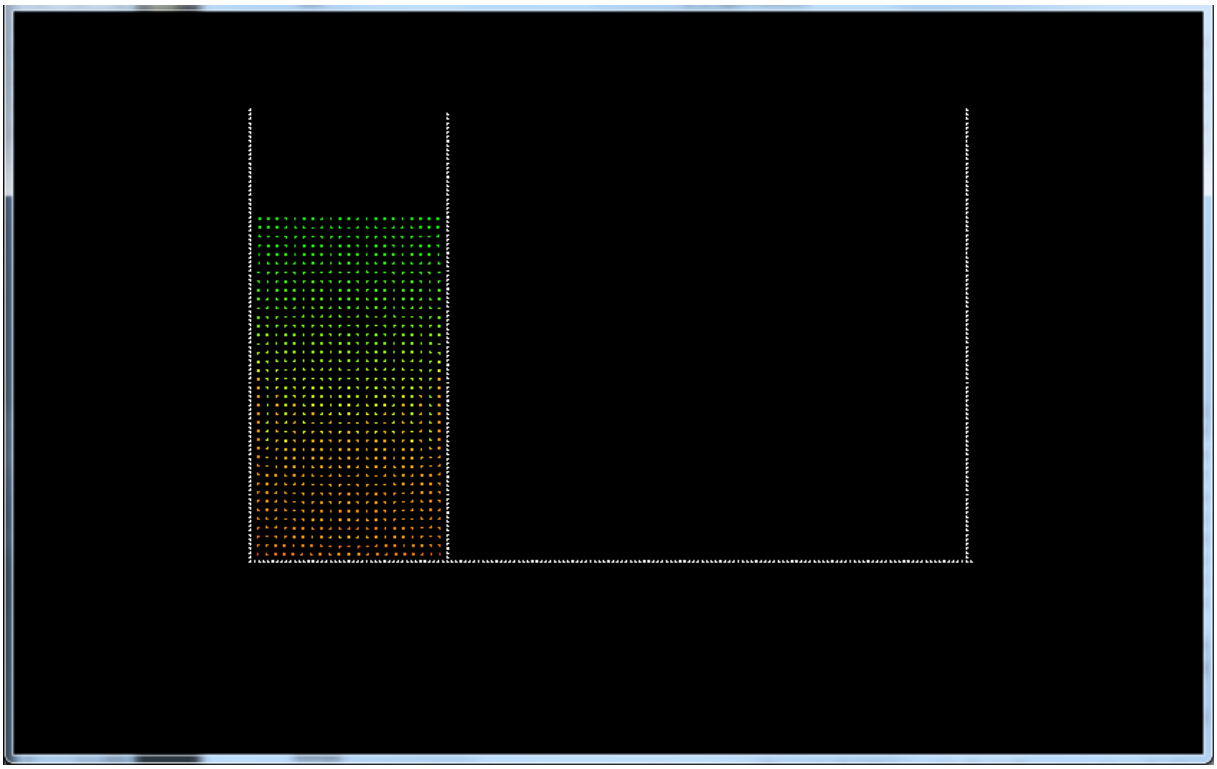


Figure 16 – visualization of the pressure as the water have reached the bottom and side walls

It is seen that the pressure rises at the impact between the bottom particles and the “wall”. The same is seen when the particles interacts with the side walls.

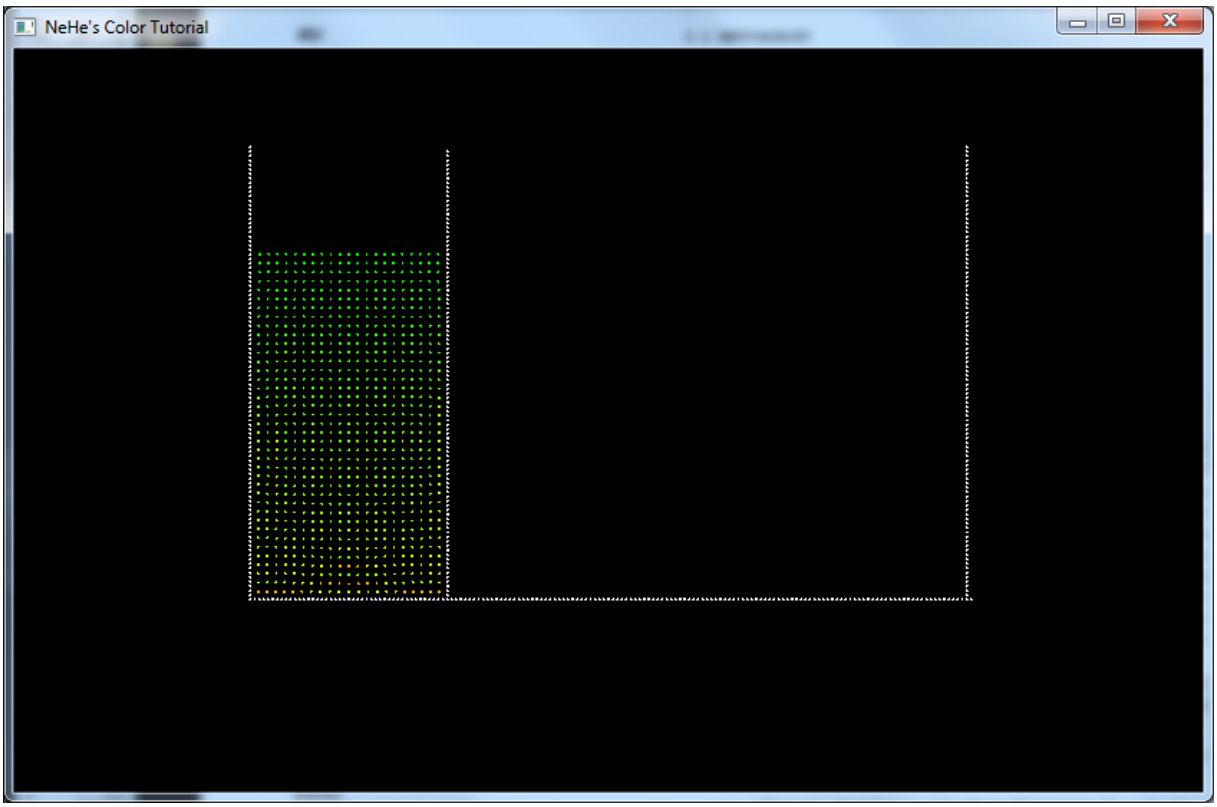


Figure 17 – The fluid at near equilibrium

In Figure 17 the fluid is close to rest. By the visualization it may be observed that the surface is calm, the pressure increases with the depth, and that the particles rest against the “walls”. A close look at the side walls shows that the particles at the bottom are closer than the one at the surface. Thus is due to the boundaries give a repulsive force as a particle comes close enough, hence the particles at the bottom with high pressure will achieve equilibrium closer to the boundaries than the ones at the surface.

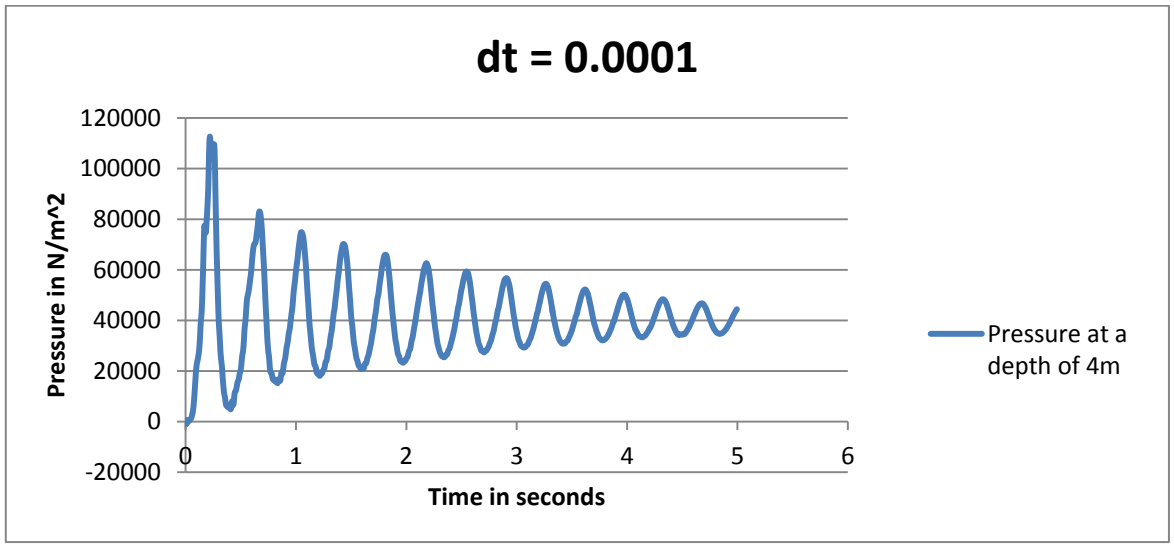


Figure 18 – Pressure at a depth of 4 metres for calming water

In the simulation the pressure at the bottom were stored as results. The pressure was found by calculating the mean pressure of the particles at a depth between 3.7 and 4.0 metres. The hydrostatic pressure at this depth is expected to be $40.000 N/m^2$calculated from the analytical solution $\rho g h$. Figure 19 displays the result from the simulation. The pressure starts of at zero which is the initial value and reaches as high as $11 * 10^4 N/m^2$. The sudden increase in pressure is because the fluid is initially placed a small distance above the bottom. The pressure oscillates as the system goes towards equilibrium. The asymptotic value seems to be the correct analytical value of $4'10^4 N/m^2$. However the time spent before this system reaches equilibrium can be questioned. Most likely the system will not reach a state of full equilibrium because of small errors which give the system new energy. However the calm surface, nice behavior of the particles and the asymptotic value are good results.

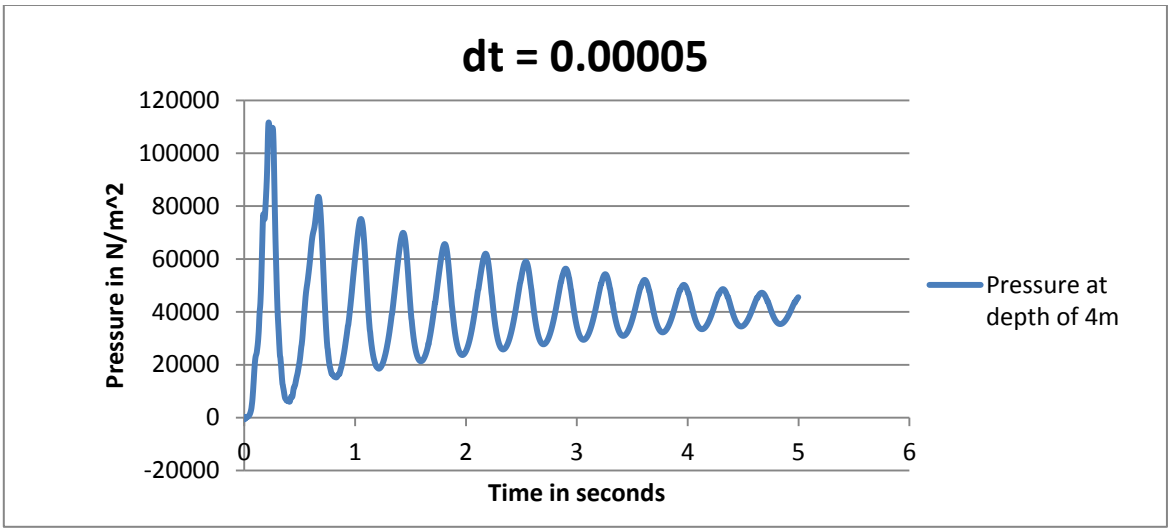


Figure 19 – Pressure at a depth of 4 metres for calming water. Smaller time steps.

A new simulation was made with a time step half of that in the previous one. This was to investigate if the time integration was adding energy to the fluid. The results turned out just the same, as can be seen in Figure 19. An interesting finding was however made when less particles were used.

Breaking dam

The last case implemented and tested is the "broken dam" scenario. A volume of fluid, in this case water, is initially placed inside a container. At a given time, one of the walls is removed and the fluid is free to move in one direction. An illustration of this can be seen in the figure below where the fluid is free to move in the positive x-direction. After flowing a given distance, the flow encounter a vertical wall. This case will partly test the method's ability to adept for large deformations.

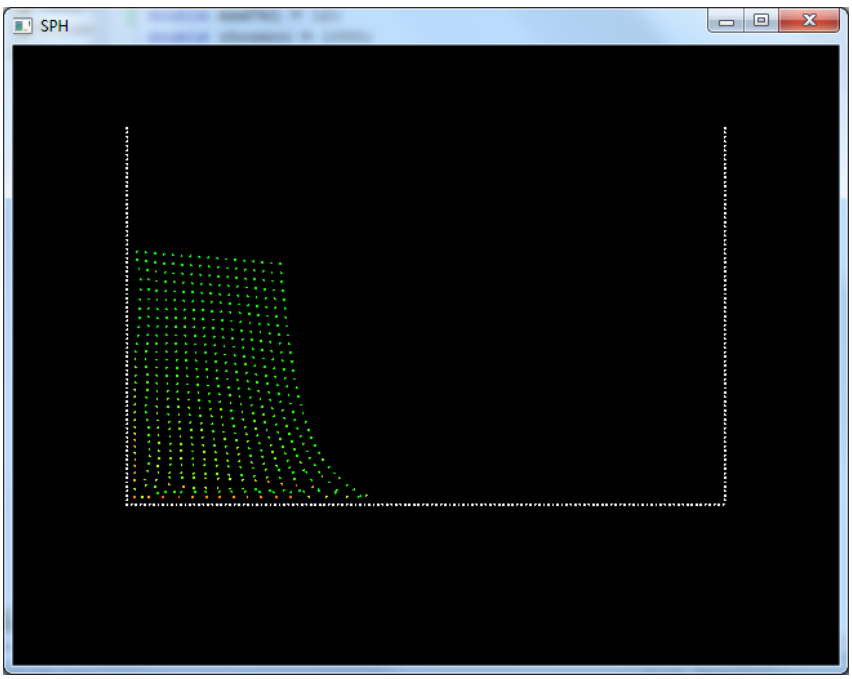


Figure 20 – "The braking dam" shortly after start of simulation

The dimension of the dam is an initial water height of 40 metres and width of 20 metres. The opposite wall is placed 60 metres from the fluid. The pressure of the braking dam simulation has not been measured. The visualization gives reason to believe that the pressure results would not be very useful. The pressure varies quite much from one particle to the next, hence not as uniform as desired.

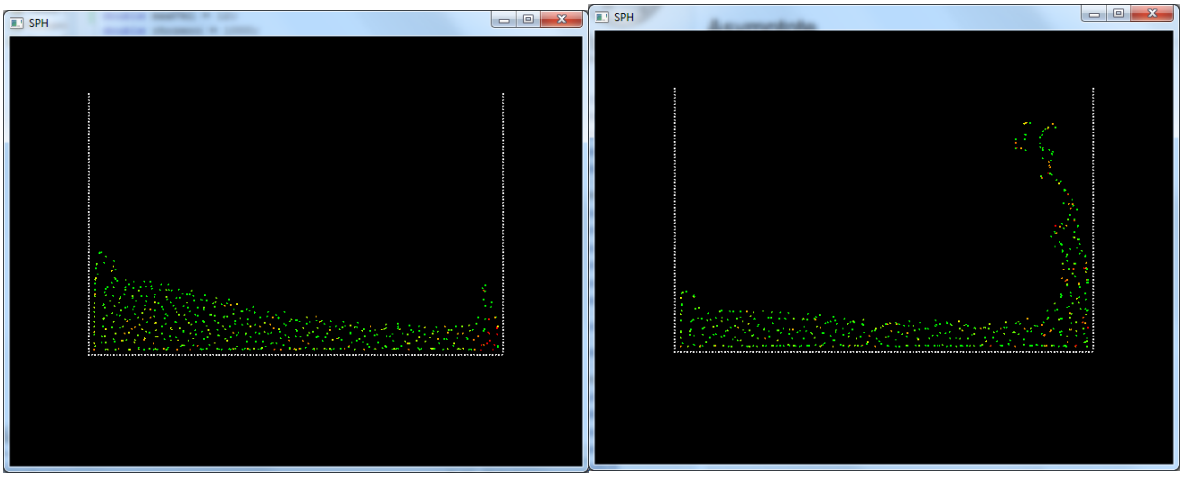


Figure 21 – Impact of wave against opposite wall

However if the pressure was averaged over the area of interest there could seem to be some reasonable good results. This could for example be at the impact of the wave against the opposite vertical wall as shown in Figure 21.

Further work

The results achieved by using the implemented code shows good potential. However the best steps to improve this is to give better results would be

- To implement a time integration function with variable time step
- Implement a fast neighbor search
- Implement support for multi core calculations
- Make use of GPU or program entire solution on GPU
- Parameter analysis
- Try various modification/improvements such as repulsive term etc.
- Optimizing the code with respect to memory performance etc.

Concluding remarks

Smoothed Particle Hydrodynamics is in many problems a very suitable method, especially those with large deformations and free surfaces. Simulations of large impacts and multiphase flows may be a strong area as well. The interpolating functions are easy to implement and a SPH program does not require a huge amount of programming code to function. Though to achieve good results for more complex problems many of the simple SPH approximations should be modified. For example there

have been constructed many smoothing functions and to choose there should be taken into account factors such as computational cost, numerical stability, approximation to the Dirac Delta function, differentiability, compact support and more. There are several approximations for the density such as the summation approach and continuity approach, the continuity density approach appears to be better than the summation approach due to its computational cost advantage and its applicability at free surfaces. The boundary conditions have been modelled several ways as for example boundary particles, virtual particles on the boundary and in the region boundary. Several approximations for velocity and energy have been used. Other improvement exists such as the XSPH variant which prevents particles from penetrating each other and makes particles more ordered. For simulating real fluids such as water, viscosity should be implemented. The SPH method might be a mature method, but it seems as if there still is quite much research left. It is hard to find a general common opinion or documentation of which approaches are the best. All this implies that when implementing a program based on SPH there are many factors to consider for achieving the best results and highest efficiency.

The Smoothed Particles Hydrodynamics method has been implemented to simulate three test cases. It has been shown that the evolution of the axis of a drop with an initial velocity can be approximated with only a few percent errors. The pressure at the center of the drop is however not as close to the analytical solution as desired, but shows some potential.

In addition the method's stability and ability to reproduce hydrostatic pressure is somewhat tested by simulating a volume of water placed in a container. The pressure at a given depth is measured as the fluid reaches equilibrium. Results show a pressure which are oscillating around a value equal to the analytical pressure. The surface is calm, the pressure oscillation decreases but does not vanish and the method shows no instability in this case.

A broken dam is investigated as the last scenario. The deformation of the fluid gives an impression to be realistic. The pressure results are not given as numbers, but can to some extend be seen in the visualization. The pressure distribution shows large differences in neighboring particles, thus seem a bit off. However it is noticed that the pressure rises at important areas such as at the impact against the opposite wall.

It is found that a relatively basic implementation of the method gives approved results in terms of deformations. The results in pressure are not as great, but show potential. This implies that by some further work the SPH method has great potential in simulating suitable scenarios.

References

Batchelor, G. K. (1973). *An introduction to fluid dynamics.* Cambridge, UK.

Jiannong Fang, A. P. (2008). Improved SPH methods for simulating free surface flows.

Liu, G. R. (2003). *Smoothed Particle Hydrodynamics : A Meshfree Particle Method.* World Scientific Publishing Company, Incorporated.

Monaghan. (1994). Simulating free surface flows with SPH.

Pákozdi, C. (2005). *Smoothed Particle Hydrodynamics.*

Schlatter, B. (1999). *A Pedagogical Tool Using Smoothed Particle Hydrodynamics to Model Fluid Flow Past a System of Cylinders.*