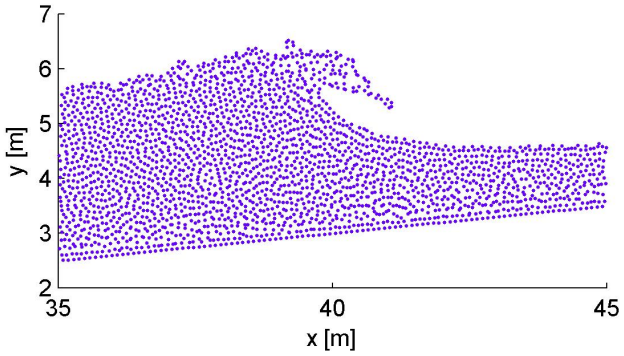
Report MSc Thesis

Smoothed Particle Hydrodynamics

A Study of the possibilities of SPH in hydraulic engineering



Lynyrd de Wit
April 2006

Delft University of Technology
Faculty of Civil Engineering and Geosciences
Environmental Fluid Mechanics Section

TU Delft
Technische Universiteit Delft

MSc Thesis
Lynyrd de Wit, April 2006

Smoothed Particle Hydrodynamics

A Study of the possibilities of SPH in hydraulic engineering

Thesis Committee:
prof.dr.ir. G.S. Stelling
prof.dr.ir. A.W. Heemink
dr.ir. M. Zijlema
ir. R.J. Labeur

Preface

During the International symposium coastal hydro- and morphodynamics in honour of Prof. Jurjen Battjes in May 2004 one of the speakers was Robert Dalrymple. He presented some results of numerical modelling of waves using the SPH method. The innovative way to describe water with particles attracted my attention immediately. Numerical modelling of water motion is one of the most interesting parts of Civil Engineering for me. I was happy to get the challenging graduation subject to study the possibilities of SPH in hydraulic engineering.

I'd like to thank my enthusiastic committee members. Robert Jan Labeur was stimulating and always ready to help me to find solutions for the problems we faced. Prof. G.S. Stelling was very interested in the capabilities of SPH. M. Zijlema could give good help with programming in Fortran and numerical integration. A.W. Heemink was interested as an external member of my committee. I'm thankful to all members for their thorough feedback in the meetings.

Most of all I'm thankful for the capacities God gave me to fulfill this wonderful graduation work, and for the perspective He gives me every day. Solving differential equations is very nice, but in the end life is about relations with each other and with Him.

Lynyrd de Wit

Abstract

In this thesis the Smoothed Particle Hydrodynamics (SPH) method is described and demonstrated for some simulations in the application field of hydraulic engineering. SPH is a Lagrangian particle method which uses a smoothing function to find spatial dependencies between particles. The particles carry all the quantities and no grid is needed. Because the particles can move themselves the total material derivative of quantities can be used, this avoids problems with advective transport. SPH can be used to solve the non-hydrostatic Navier-Stokes equations to model water motion. The fluid in SPH is made slightly compressible (density variations maximal 1 %) to get a time derivative of the density and calculate the pressure explicitly from the density. The fluid is not approximated as incompressible because an implicit Poisson equation for the pressure needs to be solved then which is cumbersome to do in a particle method. Viscous stress terms are simulated with an artificial viscosity. The artificial viscosity is also needed for numerical stability. The free surface in SPH simply is the transition between an area with particles and an area without. Fast varying or intersecting surfaces like in overtopping waves are no problem to model. Boundaries are modelled with boundary particles with a repulsive force for all fluid particles coming close. Ghost fluid particles are created outside of a boundary to prevent artificial boundary effects.

A 2D version of SPH is programmed and used in several simulations from the hydraulic engineering practice. Some viscosity benchmark problems showed that the artificial viscosity approach indeed works like a viscous stress term. Velocity differences are decreased by exchange of momentum, not completely correct but in an artificial way. The SPH results for a broken dam problem and a bore at a wall are very close to experimental or theoretic results. A standing wave simulation showed that the present version of SPH suffers from a positive numerical phase error, and big dissipation from the artificial viscosity that is needed for stability. SPH gives realistic results for spilling or plunging waves on a beach. Phenomena like shoaling, wave set-down before breaking and wave set-up after breaking are reproduced correctly by SPH. Wave overtopping and breaking in the case of plunging waves is found from the Navier-Stokes equations without further assumptions. Finally SPH is used to model the flow over a sharp weir. SPH can handle the sharp pressure gradients and the water jet at the weir. The discharge is close to the expected theoretical discharge.

The simulations in this thesis show that SPH can be used in many different situations in hydraulic engineering. Especially in problems with large pressure gradients, fast varying

water levels and intersecting free surfaces the advantages of SPH show to full extend. Unfortunately a small time step is needed for stability, together with many particles needed for enough resolution, this leads to considerable calculation times. The application of SPH is therefore restricted to local and short phenomena. SPH can be used in situations where many other methods fail, for instance wave overtopping can be simulated in great detail. Although SPH is a relatively new technique in the field of hydraulic engineering, it's future looks promising.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Hydraulic engineering is defined as the branch of civil engineering dealing with the use and control of water in motion. Numerical simulation is an important tool to understand the motion of water. The position of the free surface is difficult to predict when it changes rapidly in time. A multiply defined free surface as in overtopping waves is harder or even impossible to predict for many operational numerical methods. The subject of this report is Smoothed Particle Hydrodynamics (SPH), it is a particle method using a smoothing function for interactions between particles. It can be used to model water motion and claims to be a robust technique for many dynamic problems in hydraulic engineering. Fast varying water levels, difficult interaction between water and moving objects, and spectacular overtopping waves can be modelled with SPH directly from the Navier-Stokes equations, without assumptions over what will happen beforehand. This report will investigate whether the claims are true. An introduction to numerical modelling and the position of SPH among other methods is not given in this general introduction, but in chapter 2.

1.1 Problem statement

This graduation thesis will investigate the possibilities of SPH in hydraulic engineering. The aspects considered are:

1. How does SPH model water motion? Which assumptions are made and how are the equations, governing water motion, approximated in SPH?

2. Develop or find a working SPH computer code.

3. Use SPH to carry out some simulations in the field of hydraulic engineering. How do the results compare with experiments or theory? How accurate is SPH?

4. Finally, what is the conclusion about the possibilities of SPH in hydraulic engineering? What are strong and weak points? Which improvements are needed?

1.2 Outline of the thesis

After this introduction the position of SPH among other numerical approaches in hydraulic engineering is explained in chapter 2. The basic principles of SPH are the subject of chapter 3. The way functions are approximated in SPH is explained, and then applied to the equations governing fluid motion. To use SPH in a computer code, and to improve the accuracy, some implementations are needed, like time-integration, boundaries, etc. These implementations are standing in chapter 4. With the SPH equations and implementations a computer code is developed, the structure of it can be found in chapter 5. The simulations carried out with the SPH computer code are presented in chapter 6. All simulations are from the field of hydraulic engineering. Finally in chapter 7 conclusions about SPH are drawn and further improvements are recommended.

Chapter 2

Numerical simulation of free surface flows

This chapter gives a general introduction to numerical modelling in hydraulic engineering. The general interest in hydraulic engineering lies in problems with a free surface. After the introduction to numerical simulation some different approaches to model free surface flows on a computer with their key assumptions are given. This chapter does not pretend to be complete about every approach, but gives a very brief overview. The position of the subject of this paper, Smoothed Particle Hydrodynamics, among the other approaches will become clear.

2.1 Introduction to numerical simulation

Numerical simulation has become an important tool for solving engineering problems. With the help of increasing computer power less and less assumptions are necessary and problems can be solved with more details. Numerical simulations are replacing expensive and difficult experiments in laboratories more and more. To carry out a numerical simulation on a computer a physical problem is translated into a discrete set of mathematical formulas. For fluid motion this gives a set of partial differential equations (PDE's) in time and space. Pressure, and a velocity component for every used dimension, are the only main independent field variables. With enough initial and boundary conditions the PDE's can be solved giving the pressure and velocity at every point, at every time. In most cases analytical solutions are not available. But when space is divided in a finite number of components, and time is divided in a finite number of steps, the solution of the PDE's can be found by numerical integration. Many ways to discretise the PDE's and to handle a free surface are possible. They all have advantages and disadvantages for certain problems. Four approaches which are able to model free surface flows will be introduced below. These are grid based methods, methods combining a grid with particles, a method combining a grid with a surface finder, and particle methods without grid.

2.2 Grid based methods

In grid based numerical methods space is divided in a finite number of cells forming a grid. The PDE's governing fluid motion are solved on the grid. In most methods the grid is fixed and the fluid is flowing through it, this is called an Eulerian grid. At the centre of a cell the pressure is calculated, at the boundary of a cell the velocities are calculated giving fluxes between the cells. Grid based numerical methods can be divided in two main approaches. One approach is using the finite differencing method (FDM), space is discretised with an orthogonal grid leading to a structured grid. The approach called finite volume method (FVM) is practically the same as FDM. The spatial derivatives of the original PDE's are approximated on the grid. In the limit of a grid with an infinite number of cells the discretised spatial derivative will give the original analytical derivative. By this discretisation the original PDE's are converted into a set of ordinary differential equations (ODE's) in time, with a numerical time integration scheme the solution can be progressed in time. On a structured grid the discretisation is straight forward and conservation of mass is easily obtained. The drawback is that real world is not orthogonal and structured. The other approach using a grid is called the finite element method (FEM), this method divides space in a finite number of elements. These elements are often triangles, but other elements are possible as well. The original PDE's are not discretised in space on the grid, but a shape function is used to approximate the spatial solution of the original PDE's on the grid. Again the solution is progressed in time by a numerical time integration scheme. Irregular geometry can be modelled easily with FEM because of the use of triangles. FEM can be combined with a Lagrangian grid moving along with the flow, but for simulations considered in hydraulic engineering the large deformation gives problems. Both grid based methods are used often and give accurate solutions for many problems concerning free surface flows. However grid based methods do have some drawbacks. FDM and FEM can have vertical adaptive grids for a varying water level, but the water level may not change too fast in time or space. Intersecting or breaking surfaces cannot be modelled. A fixed grid also gives difficulties with the correct calculation of the transport between cells. This advective transport is non-linear and therefore hard to solve efficiently and correctly.

2.3 Grid with particles

To compensate some of the drawbacks of grid based numerical methods they have been combined with particles. One of the first methods combining particles with a grid was the Particle-in-Cell method (PIC) for compressible flows of Harlow (1963). The particles are used to handle advective transport, the other terms of the PDE's are solved on the grid. Every timestep the values from the particles and the grid are exchanged by interpolation to give a correct solution of the PDE's inclusive advection. The problems with non-linear advective transport in a grid based method is solved this way, but the exchange of information between the particles and the grid causes excessive numerical diffusion. Recently some improvements are made to PIC to handle incompressible flow and to reduce the numerical diffusion by making the particles the fundamental representation of the fluid. The grid is only used to

calculate interaction between the particles. This improves the results with PIC a lot, but it would be interesting to go one step further and calculate the interaction between particles directly without a grid. Another combination of particles with a grid is the Marker-and-Cell method (MAC) from Harlow and Welch (1965). This method uses a grid to solve the PDE's inclusive advective terms and marker particles to indicate the fluid configuration. The marker particles do not have fysical quantities like mass or volume and they do not participate in the calculation. They only serve as flow visualisation and indicate the free water surface. MAC can handle varying water levels and even breaking surfaces. But MAC does need a lot of particles to track a fast varying water level accurately. Calculation time and memory requirements are large, therefore MAC is not often used for 3D simulations but mostly for 2D problems only. MAC has the same problems with non-linear advection as the grid based methods without particles.

2.4 Grid with free surface finder

It would be interesting to combine the accurate surface handling capabilities of MAC without the big computational costs. A method which can achieve this is the Volume-of-Fluid method (Hirt and Nichols 1981). It uses a fixed Eulerian grid to solve the equations of fluid motion. By tracking the volume of fluid in every cell it finds the free surface, the slope of the surface is found by using the volume fractions in the neighbouring cells. The time evolution of the surface is calculated by moving the volume fractions through the grid correctly, also free surface boundary conditions are applied at the free surface. Its use of volume tracking for finding the free surface is robust enough to handle breaking surfaces. The volume fractions can vary continuously, therefore the free surface can be at any level, not only at cell boundaries. VOF can produce impressive results for free surface simulations, including for example breaking waves with splash up. A lot of administration to simulate a correct free surface is needed though, and due to its use of a fixed Eulerian grid VOF is struggling with the advective transport.

2.5 Particle methods

All currently mentioned approaches are using a fixed grid to solve the PDE's governing fluid motion. In a particle method no grid is used, but all flow quantities are carried by particles, they have mass, volume, pressure, velocity, etc. The quantities of a particle are moving along with it, they can only change by external influence or internal production, not by transport over the boundaries of the particle. Therefore there are no problems with non-linear advective transport in particle methods. The spatial derivatives are approximated by interaction between the particles. The free surface is just the transition between an area with fluid particles and an area without. Particle methods can handle fast varying, breaking or intersecting surfaces. These interesting features are the reason to further investigate the possibilities of a particle method in hydraulic engineering. The two most popular particle methods are Smoothed Particle Hydrodynamics (SPH) and Moving Particles Semi-implicit

(MPS). SPH is used for slightly compressible flow, in MPS an extra equation is solved to guarantee an incompressible flow. Unfortunately the semi-implicit equations in MPS to ensure incompressibility are cumbersome to solve numerically. SPH is chosen because it is robust, much easier to implement than MPS, and it has been used for several 2D and 3D simulations of complex free surface flows (Gómez-Gesteira and Dalrymple 2004), or breaking waves (Rogers and Dalrymple 2004).

Chapter 3

Smoothed Particle Hydrodynamics equations

3.1 The basic equations of fluid motion

The motion of fluids can be described by three conservation laws, namely conservation of mass, conservation of momentum and conservation of energy. The conservation laws are formulated under the assumption that the fluid is a continuous medium. From these conservation laws partial differential equations are derived. For the type of problems in this thesis only conservation of mass and conservation of momentum are to be considered. The lecture notes from Wesseling (2002) were very useful for this section.

3.1.1 Notation

A Cartesian coordinate system will be used with components (x, y, z). Only two dimensional problems have been tackled, but the presented equations are valid for three dimensions as well. Bold-faced small letters denote vectors, for example velocity $\mathbf{u} = (u_x, u_y, u_z)^T$. Normal-faced letters denote scalars. The total derivative of a property ϕ is denoted by $\frac{D\phi}{Dt}$. The total derivative accounts for the variation of a property when following a particular particle. It can be expressed in local field quantities by:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi. \tag{3.1}$$

3.1.2 Conservation of mass

Without production of mass, there is no change of mass in an arbitrary material volume $V(t)$. In equation form this becomes:

$$\frac{d}{dt} \int_{V(t)} \rho dV = 0, \tag{3.2}$$

with the transport theorem this can be written as:

$$\int_{V(t)} (\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u})) dV = 0. \tag{3.3}$$

This is valid for every volume $V(t)$, so the integrand must be zero:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \rightarrow \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{u} = 0 \rightarrow$$
$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u}. \tag{3.4}$$

Without the influence of salinity or temperature on the density, the variations of the density in water are extremely small. Often water is approximated as a totally incompressible fluid, because the equations for an incompressible fluid are easier to solve than for an almost incompressible fluid. In an incompressible fluid the density of each material particle remains constant:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u} = 0. \tag{3.5}$$

From this it follows that $\nabla \cdot \mathbf{u} = 0$, an incompressible fluid is divergence free. In an incompressible fluid there is no relation between pressure and density, because the density is always the same while the pressure can vary. The pressure is calculated implicitly by solving a Poisson equation. For SPH this implicit way of calculating the pressure is very cumbersome, therefore another approach is chosen. The almost incompressible medium water is not made totally incompressible but more compressible in SPH. The full mass conservation equation has to be used, reading:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{u} = -\nabla \cdot (\rho \mathbf{u}) + \mathbf{u} \cdot \nabla \rho. \tag{3.6}$$

This equation gives a time derivative of the density. With a relation between the pressure and the density it is no longer necessary to solve the implicit Poisson equation for the pressure. When atmospheric pressure is neglected, the relation between pressure and density is given by the following equation of state (see also Batchelor (1974)):

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right], \tag{3.7}$$

where p is the pressure in Pa, $\gamma = 7$, $B = c^2\rho_0/\gamma$, $\rho_0 = 1000kg/m^3$, and c is the used speed of sound. When the real sound speed in water of 1480 m/s is used in the constant B, this relation agrees with measured data for water within a few percents, for pressures less than $10^{10}Pa$. With a large c the time step has to be small for stability reasons, this leads to large calculation times. In stead of using the real sound speed an artificial sound speed of $c \approx 10u_{max}$ is used in SPH. The variation in the density of a fluid is given by Monaghan (1994):

$$\frac{\Delta \rho}{\rho} \sim \frac{u^2}{c^2} = M^2, \tag{3.8}$$

where u is a typical fluid velocity, c is the speed of sound in water and M is the Mach number. The relative density differences are proportional to M^2. With $c \approx 10u_{max}$ the maximum relative density differences are small, order $\sim 1\%$. Now the calculation time stays reasonable and water is only slightly compressible in SPH.

3.1.3 Conservation of momentum

Conservation of momentum implies that the rate of change of momentum of a material volume is equal to the total force on the volume. The total force exists of surface forces proportional to the surface $dS(t)$ of a volume, and body forces proportional to its mass $\rho dV(t)$.

$$\frac{d}{dt}\int_{V(t)} \rho \mathbf{u} dV = \int_{S(t)} \mathbf{f}_s dS + \int_{V(t)} \rho \mathbf{f}_b dV. \tag{3.9}$$

The surface tension $\mathbf{f}_s$ can be expressed with an inner product of a second order tensor $\mathbf{T}$ with the outward unit normal vector $\mathbf{n}$ on S.

$$\int_{S(t)} \mathbf{f}_s dS = \int_{S(t)} \mathbf{T} \cdot \mathbf{n} dS = \int_{V(t)} \nabla \cdot \mathbf{T} dV. \tag{3.10}$$

Where the last step is known as the divergence theorem. Combining equation 3.9 with 3.10 and using the transport theorem together with conservation of mass (equation 3.6) to rewrite the first term in equation 3.9 this becomes:

$$\int_{V(t)} \rho \frac{D\mathbf{u}}{Dt} dV = \int_{V(t)} \big(\nabla \cdot \mathbf{T} + \rho \mathbf{f}_b\big) dV. \tag{3.11}$$

Since this holds for every volume $V(t)$, it results in:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \mathbf{T} + \rho \mathbf{f}_b. \tag{3.12}$$

The LHS represents the rate of change of momentum of a material volume. $\mathbf{T}$ is the second order stress tensor representing the surface forces, $\rho \mathbf{f}_b$ represents the body forces. With a constitutive relation for a Newtonian fluid to relate the stress tensor to the motion of fluid, conservation of momentum yields the Navier-Stokes equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}_b, \tag{3.13}$$

where the LHS is the rate of change of momentum. The first term on the RHS is a pressure gradient, second term represents the influence of viscosity, last term is an acceleration representing the body forces. An example of $\mathbf{f}_b$ is the influence of gravity.

3.2 Functions and derivatives in SPH

In SPH the fluid is discretised in large particles, they have the properties: mass, density, pressure, position and velocity. These properties are attributed to the centre of each particle. The particles with their properties are scattered in space, and can move independently. The key idea in SPH is that it uses a smoothing function to produce smooth, continuous interpolation fields of fysical properties from the discrete particle information in the computational domain.

3.2.1 Functions in SPH

A continuous function $f(\mathbf{x})$ can be approximated by the integral interpolant:

$$\langle f(\mathbf{x})\rangle = \int_\Omega f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}'. \tag{3.14}$$

Where $W(\mathbf{x}-\mathbf{x}',h)$ is an interpolation kernel with h as the smoothing length, which determines the width of the kernel. Section 3.3 will say more about smoothing kernels. When space is not continuous anymore but discretised into a set of particles the approximation of a function becomes:

$$\begin{aligned}\langle f(\mathbf{x})\rangle &= \sum_j f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j,h)\Delta V_j,\\ \langle f(\mathbf{x})\rangle &= \sum_j f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j,h)\frac{1}{\rho_j}(\rho_j\Delta V_j),\\ \langle f(\mathbf{x})\rangle &= \sum_j \frac{m_j}{\rho_j}f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j,h).\end{aligned} \tag{3.15}$$

When a function needs to be known at a particle position $\mathbf{x}_i$ this approximation can be written as:

$$\langle f(\mathbf{x}_i)\rangle = \sum_j \frac{m_j}{\rho_j}f(\mathbf{x}_j)W_{ij}, \tag{3.16}$$

where i,j are used as particle-indexes, m_j and ρ_j are respectively mass and density belonging to particle j, $W_{ij}=W(\mathbf{x}_i-\mathbf{x}_j,h)$ is the value of the smoothing kernel for the interaction between particle i and j.

3.2.2 Spatial derivatives in SPH

Following the SPH approach to describe a function $f(x)$, the spatial derivative at $\mathbf{x}$ can be approximated with:

$$\begin{aligned}\langle \nabla\cdot f(\mathbf{x})\rangle &= \int_\Omega [\nabla\cdot f(\mathbf{x}')]W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}',\\ \langle \nabla\cdot f(\mathbf{x})\rangle &= \int_\Omega \nabla\cdot[f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)]d\mathbf{x}' - \int_\Omega f(\mathbf{x}')\cdot\nabla W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}'.\end{aligned} \tag{3.17}$$

The first integral of equation (3.17) can be converted into a surface integral using the divergence theorem:

$$\langle \nabla\cdot f(\mathbf{x})\rangle = \int_S [f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)]\cdot\mathbf{n}dS - \int_\Omega f(\mathbf{x}')\cdot\nabla W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}', \tag{3.18}$$

where $\mathbf{n}$ is the outward unit vector normal to surface S. The smoothing function has compact support, which means that $W(\mathbf{x}-\mathbf{x}',h)=0$ when $|\mathbf{x}-\mathbf{x}'|>\kappa h$. When a spatial derivative is needed at a point more than κh away from the surface S the first term in equation 3.18 is zero and the spatial derivative of a function becomes:

$$\langle \nabla\cdot f(\mathbf{x})\rangle = -\int_\Omega f(\mathbf{x}')\cdot\nabla W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}'. \tag{3.19}$$

For derivatives at positions within κh from surface S adaptations have to be made to prevent artificial boundary effects. With the steps from equation 3.15 the derivative can be written in discrete formulation, giving

$$\langle \nabla \cdot f(\mathbf{x}) \rangle = -\sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) \cdot \nabla W(\mathbf{x} - \mathbf{x}_j, h). \tag{3.20}$$

A spatial derivative of a function at a particle position $\mathbf{x}_i$ can be written as:

$$\langle \nabla \cdot f(\mathbf{x}_i) \rangle = -\sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) \cdot \nabla_j W_{ij}, \tag{3.21}$$

where

$$\nabla_j W_{ij} = \frac{\mathbf{x}_j - \mathbf{x}_i}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}}. \tag{3.22}$$

Here $\mathbf{x}$ is the vector containing the position of a particle and r_{ij} is the absolute value of the distance between particle i and j. Furthermore$\nabla_j W_{ij} = -\nabla_i W_{ij}$ and equation 3.21 can also be written as:

$$\langle \nabla \cdot f(\mathbf{x}_i) \rangle = \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) \cdot \nabla_i W_{ij}. \tag{3.23}$$

Now the derivative does not have a negative sign on the RHS because the gradient is taken with respect to particle i.

3.3 Kernel

3.3.1 Properties of a kernel

SPH is not using a grid but particles to discretise space. In the last section it is shown that a smoothing function is used to approximate a function and its derivative from the values known at particle positions. The choice of the smoothing kernel together with a adequate smoothing length is therefore important for both accuracy and speed. First some important properties of kernels are summed up (Liu and Liu 2003).

1. The smoothing function must be unity over its support domain:

$$\int_\Omega W(\mathbf{x}) d\mathbf{x} = 1. \tag{3.24}$$

2. The smoothing function must have compact support, which means:

$$W(\mathbf{x} - \mathbf{x}', h) = 0, \text{ for} |\mathbf{x} - \mathbf{x}'| > \kappa h, \tag{3.25}$$

   where κh defines the influence domain where contributions of other particles cannot be neglected. The scaling factor κ normally is set to 2.

3. When the smoothing length h goes to zero, the smoothing function should behave like a Dirac delta function:

$$\lim_{h \to 0} W(\mathbf{x} - \mathbf{x}', h) = \delta(\mathbf{x} - \mathbf{x}'). \tag{3.26}$$

4. The smoothing function must be positive in its entire domain.

5. The smoothing function must be maximum when $|\mathbf{x}-\mathbf{x}'| = 0$, and zero when $|\mathbf{x}-\mathbf{x}'| = \kappa h$. In between the smoothing function must decrease monotonically. Particles close to each other must have more influence on each other than particles with a larger distance.

6. The smoothing function must be symmetric. Two particles at the same distance of a particle i, but at different position must have the same influence on particle i.

7. The smoothing function should be sufficiently smooth. A smoother function is less sensitive to particle disorder.

3.3.2 Piecewise cubic spline

Every function which meets the properties summed up above can be used as smoothing kernel. A Gaussian kernel ($W_{ij} = \alpha_d e^{-q^2}$, see equation 3.27 for explanation of the terms) can fulfill six of the seven properties excellently. Unfortunately it does not have compact support. A smoothing function which looks like a Gaussian kernel, but also has compact support is the Piecewise cubic spline. It is used a lot in SPH practice, for instance see (Monaghan 1994) and (Liu and Liu 2003). The Piecewise cubic spline is used for the simulations in this report, it is defined by:

$$W_{ij} = \alpha_d \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \leq q < 1 \\ \frac{1}{4}(2-q)^3 & 1 \leq q \leq 2 \\ 0 & q > 2 \end{cases} \tag{3.27}$$

where α_d is a scaling factor making the smoothing function unity as mentioned in equation 3.24. For 1D, 2D and 3D problems α_d is $2/(3h), 10/(7\pi h^2)$ and $1/(\pi h^3)$ respectively. W_{ij} is not dimensionless, it has the dimension (length)$^{-2}$ for 2D problems. $q = r_{ij}/h$, r_{ij} is the absolute distance between particles i and j. W_{ij} depends on the smoothing length h and via q it depends on distance r_{ij}. In figure 3.1 left the Piecewise cubic spline is plotted for two dimensions with $h = 1$ m.

In order to get a spatial derivative of a function the analytical spatial derivative of the kernel is needed, see equation 3.23. The analytical spatial derivative of the Piecewise cubic spline is given by:

$$\nabla_i W_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}}, \tag{3.28}$$

with

$$\frac{\partial W_{ij}}{\partial r_{ij}} = \alpha_d \begin{cases} -\frac{3}{4}\frac{1}{h}(4q - 3q^2) & 0 \leq q < 1 \\ -\frac{3}{4}\frac{1}{h}(2-q)^2 & 1 \leq q \leq 2 \\ 0 & q > 2 \end{cases} \tag{3.29}$$

All used variables are given in equation 3.27. $\frac{\partial W_{ij}}{\partial r_{ij}}$ has dimension (length)$^{-2}$/length for 2D problems. In figure 3.1 (right) $\frac{\partial W_{ij}}{\partial r_{ij}}$ of the Piecewise cubic spline is plotted for two dimensions, again with $h = 1$ m. $\frac{\partial W_{ij}}{\partial r_{ij}}$ is always negative. From equation 3.28 it follows that

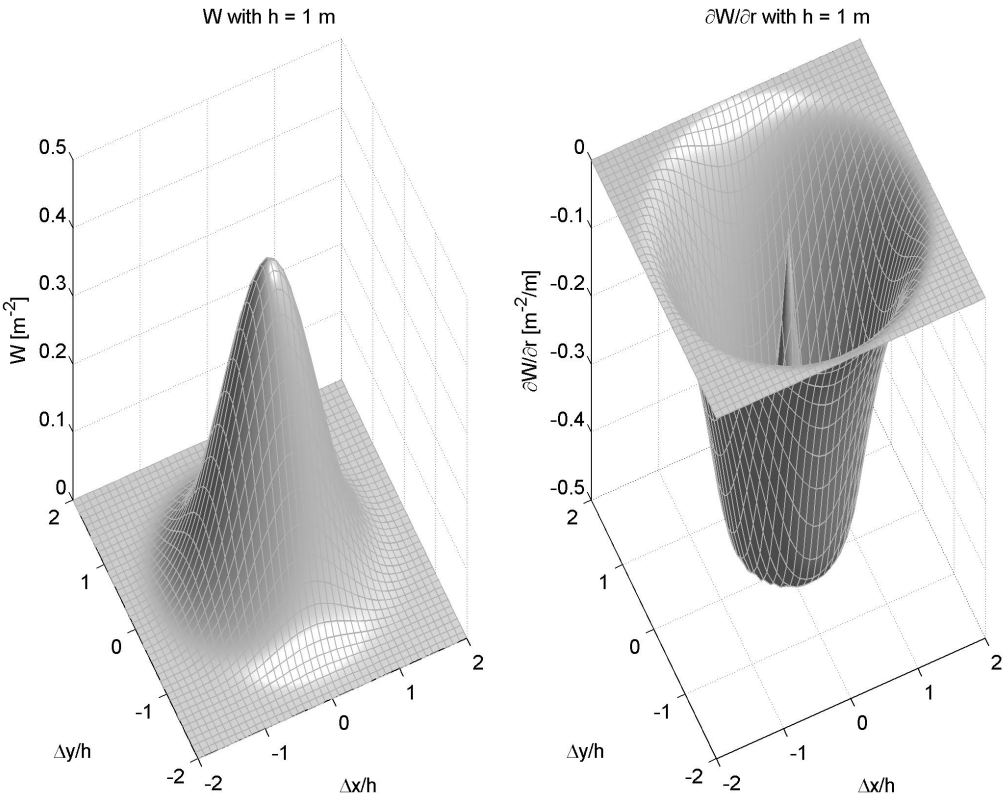


Figure 3.1: Piecewise cubic spline in two dimensions (left), spatial derivative of the Piecewise cubic spline (right).

the corresponding $\frac{\partial W}{\partial x}$ is negative when $x_i - x_j > 0$ and positive when $x_i - x_j < 0$. The same is true for $\frac{\partial W}{\partial y}$ in y direction. This can be seen in figure 3.2.

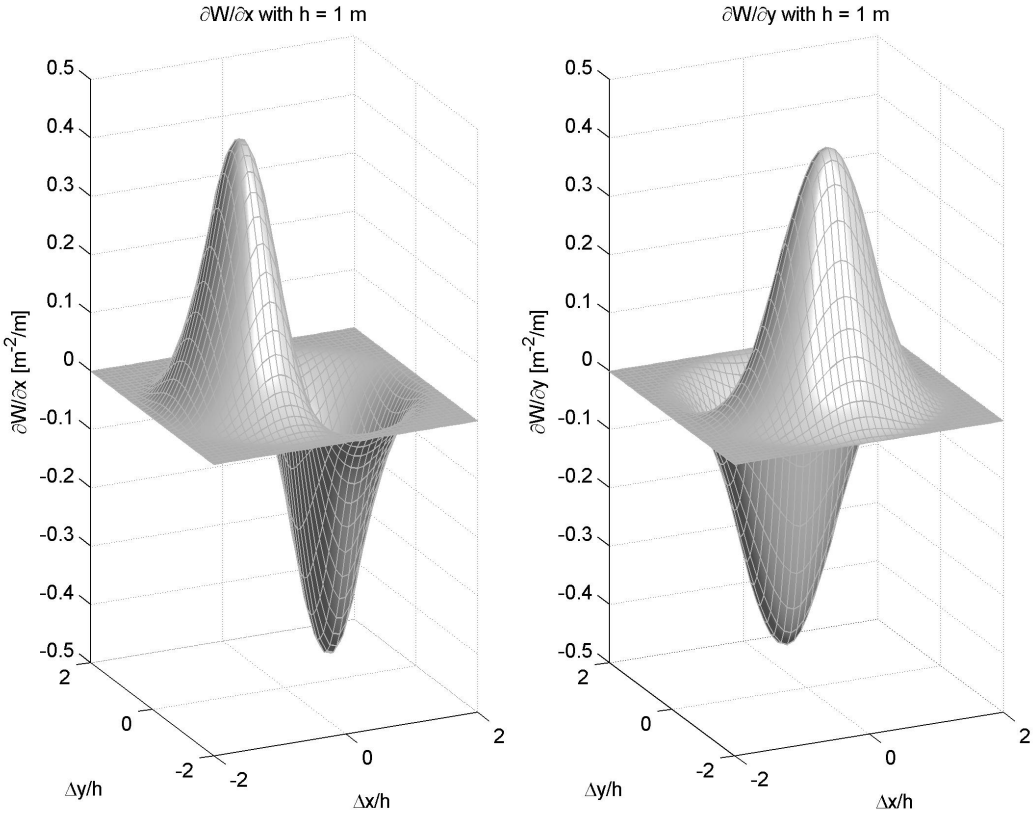


Figure 3.2: Piecewise cubic spline spatial derivative in x (left) and y (right) direction.

3.3.3 Smoothing length

As can be seen in figure 3.1 of the Piecewise cubic spline there is no interaction when $r_{ij}/h > 2$, because then W_{ij} and $\frac{\partial W_{ij}}{\partial r_{ij}}$ are zero. The choice of the smoothing length h is determining the number of interactions for each particle. When h is too small, there are not enough particles nearby to interact with, giving a low accuracy. When h is too big, local properties are smeared out too much, this gives a low accuracy again and the calculation becomes slow. With a smoothing length around 1.4 times the initial particle spacing Δr, the circle of influence has a radius of $2.8\Delta r$. In figure 3.3 it is visible that with this smoothing length every particle has 20 neighbours in two dimensions. That is a good optimum for both accuracy and calculation speed. Because the particles have the tendency to keep their neighbours at the initial distance Δr during the simulation, the number of neighbours for each particle remains fairly constant.

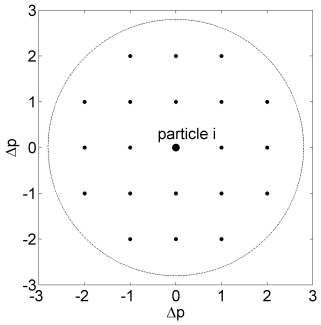


Figure 3.3: Influence domain of particle i with $h = 1.4\Delta r$.

3.4 Equations of fluid motion in SPH

The general conservation laws for fluid motion from section 3.1 can be combined with the SPH approximations of functions from section 3.2. Together with some extra relations a closed set of equations is obtained to model fluid motion. The equations are derived following Monaghan (1994) and are used by many other researchers modelling with SPH. For every equation the interaction between particles is made symmetric to be sure that the influence of particle i on j is equal to the influence of j on i.

3.4.1 Conservation of mass

The conservation of mass equation derived in section 3.1.2 reads:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{u} = -\nabla\cdot(\rho\mathbf{u}) + \mathbf{u}\cdot\nabla\rho. \tag{3.6}$$

When the particle approximation of a derivative is used (equation 3.23) this can be written as:

$$\begin{aligned}
\frac{D\rho_i}{Dt} &= -\nabla_i\cdot(\rho\mathbf{u}) + \mathbf{u}_i\cdot\nabla_i(\rho),\\
\frac{D\rho_i}{Dt} &= -\sum_j \frac{m_j}{\rho_j}(\rho_j\mathbf{u}_j)\cdot\nabla_i W_{ij} + \mathbf{u}_i\cdot\sum_j\frac{m_j}{\rho_j}\rho_j\nabla_i W_{ij},\\
\frac{D\rho_i}{Dt} &= \sum_j m_j(\mathbf{u}_i-\mathbf{u}_j)\cdot\nabla_i W_{ij}.
\end{aligned} \tag{3.30}$$

In this approach the density of a particle only changes when it is in relative motion with other particles. Note that the density of a particle also could be calculated from the mass of neighbouring particles with equation 3.16. But then a particle near the free surface would miss the contribution of particles above the free surface and the density would become too low. The calculation of the time derivative of the density does not lead to densities that are too low at the free surface.

Equation of state

From the density of each particle the pressure can be found with an equation of state (equation 3.7). The equation of state in particle form becomes:

$$p_i = B\left[\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right], \tag{3.31}$$

where $B = c^2\rho_0/7$, $\rho_0 = 1000kg/m^3$. c is the used speed of sound in the model $c \approx 10u_{max}$, with u_{max} as the maximum fluid velocity in the model. When a simulation needs to start with hydrostatic pressure, the initial density is adjusted to produce hydrostatic pressure from the equation of state (equation 3.31). Then the initial density is calculated with:

$$\rho_i = \rho_0\left(1 + \frac{\rho_0 g(d-y_i)}{B}\right)^{1/7}, \tag{3.32}$$

where d is the initial water level, y_i is the vertical particle position and the other parameters are the same as in equation 3.31.

3.4.2 Conservation of momentum

Conservation of momentum is described by the earlier derived Navier-Stokes equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f}_b, \tag{3.13}$$

First the Euler equations with only the pressure gradient on the RHS are treated. The pressure gradient could be discretized directly for particle i, leading to:

$$\begin{aligned}\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p \rightarrow \frac{D\mathbf{u}_i}{Dt} &= -\frac{1}{\rho_i}\nabla_i p, \\ \frac{D\mathbf{u}_i}{Dt} &= -\frac{1}{\rho_i}\sum_j \frac{m_j}{\rho_j} p_j \nabla_i W_{ij}.\end{aligned} \tag{3.33}$$

But now the force of particle i on j is not equal and opposite of direction to the force of j on i, because $F_i = m_i a_i$ should give $F_i = -F_j$ but $p_i \neq p_j$ and therefore:

$$-\frac{m_i m_j}{\rho_i \rho_j} p_j \nabla_i W_{ij} \neq -\frac{m_j m_i}{\rho_j \rho_i} p_i \nabla_j W_{ij}. \tag{3.34}$$

Note that $\nabla_i W_{ij} = -\nabla_j W_{ij}$. Action is not equal to reaction that is a violation of Newton's third law and will not result in conservation of linear momentum. Therefore the pressure term is adjusted to give symmetric interaction between particle i and j.

$$\frac{1}{\rho}\nabla p = \nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho, \tag{3.35}$$

because

$$\begin{aligned}\tfrac{1}{\rho}\nabla p &= \nabla\left(\tfrac{p}{\rho}\right) + \tfrac{p}{\rho^2}\nabla\rho \\ &= \tfrac{1}{\rho}\nabla p + p\nabla\left(\tfrac{1}{\rho}\right) + \tfrac{p}{\rho^2}\nabla\rho \\ &= \tfrac{1}{\rho}\nabla p + p\left(\tfrac{\partial 1/\rho}{\partial\rho}\nabla\rho\right) + \tfrac{p}{\rho^2}\nabla\rho \\ &= \tfrac{1}{\rho}\nabla p - \tfrac{p}{\rho^2}\nabla\rho + \tfrac{p}{\rho^2}\nabla\rho \\ &= \tfrac{1}{\rho}\nabla p.\end{aligned}$$

The particle approximation of the Euler equations with a symmetric pressure gradient interaction becomes:

$$\begin{aligned}\frac{D\mathbf{u}_i}{Dt} &= -\sum_j \frac{m_j}{\rho_j}\left(\frac{p_j}{\rho_j}\right)\nabla_i W_{ij} - \left(\frac{p_i}{\rho_i^2}\right)\sum_j \frac{m_j}{\rho_j}\rho_j\nabla_i W_{ij}, \\ \frac{D\mathbf{u}_i}{Dt} &= -\sum_j m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla_i W_{ij}.\end{aligned} \tag{3.36}$$

Now the particle approximation of the full Navier-Stokes equations, inclusive viscous stress terms, is given:

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}\right)\nabla_i W_{ij} + \mathbf{f}_i, \tag{3.37}$$

where the pressure gradient is discretised symmetrically, Π_{ij} is an artificial viscous stress term which is explained next section. $\mathbf{f}_i$ is the acceleration due to a body force, an example is the influence of gravity.

Artificial viscosity

Viscous stress terms could be estimated directly with the SPH approximation of a second derivative of the velocity. But the second derivative of the Piecewise Cubic spline kernel is linear in r and does change sign at $r_{ij} = 2/3h$. This would mean that for every interaction pair the direction of transfer of momentum is depending on the distance between the two particles in the interaction pair. Sometimes the transfer of momentum is from the particle with highest velocity to the particle with lowest velocity, sometimes the other way around. This is incorrect, viscosity should transfer momentum from the particle with the highest velocity to the particle with the lowest velocity independent on their distance r_{ij}. Therefore viscous terms are not estimated directly, but modelled with the following artificial viscosity term:

$$\Pi_{ij} = \begin{cases} -\frac{\alpha hc}{\bar{\rho}_{ij}} \frac{\mathbf{u}_{ij} \cdot \mathbf{r}_{ij}}{r_{ij}^2 + \varphi^2} & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0 & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} \geq 0 \end{cases} \tag{3.38}$$

where α is a constant chosen normally between following limits $1 < \alpha < 0.01$. $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$ is the average density. $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$, this is the velocity difference between two particles and $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ is the distance between two particles. $\varphi = 0.1 h_{ij}$ is used to prevent singularities when two particles have zero distance. This artificial viscosity introduces both shear and bulk viscosity, but with neglectable changes in the density it is almost entirely shear viscosity. When φ is neglected this artificial viscosity term can be understood by rewriting it to:

$$\Pi_{ij} = \begin{cases} -\frac{\alpha hc}{\bar{\rho}_{ij}} \frac{\mathbf{u}_{ij}}{\mathbf{r}_{ij}} \approx \frac{-\alpha hc}{\bar{\rho}} \frac{\Delta \mathbf{u}}{\Delta \mathbf{r}} & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0 & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} \geq 0 \end{cases} \tag{3.39}$$

The kinematic viscosity ν is proportional to αhc and $\frac{\Delta \mathbf{u}}{\Delta \mathbf{r}}$ is something like a spatial derivative of the velocity. When the artificial viscosity term is used in the full Navier-Stokes equations (equation 3.37) this leads to $\nabla(-\nu \frac{\Delta \mathbf{u}}{\Delta \mathbf{r}})$, and that does look like a viscous stress term. Monaghan (2005) related the artificial viscosity to a continuum viscosity more thoroughly. With the artificial viscosity active for both approaching and receding particles he derived that the artificial viscosity leads to $\nu \approx \alpha hc/8$. To reduce the influence of artificial viscosity in this thesis it is only active when particles are approaching, this removes the viscosity for rarefactions. It seems reasonable that it is half as effective as Monaghan derived, therefore the effective kinematic viscosity can be estimated with $\nu \approx \alpha hc/16$. This approach to model viscosity guarantees that momentum is transfered from the particle with the highest velocity to the particle with lowest velocity in an interaction pair. The artificial viscosity term Π_{ij} is a Galilean invariant and vanishes for rigid rotation.

3.4.3 Moving the particles

In stead of moving a particle with its own velocity, an average between its own velocity and the averaged field velocity from all particles nearby is taken. This is called the XSPH variant

and it prevents particle penetration. Note that every particle has two velocities now, one used to move the particle, called $\hat{\mathbf{u}}$, and one indicating the amount of momentum of the particle, called $\mathbf{u}$. The XSPH variant is defined by:

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \epsilon \sum_j m_j \left(\frac{\mathbf{u}_j - \mathbf{u}_i}{\bar{\rho}_{ij}} \right) W_{ij}, \tag{3.40}$$

$$\frac{D\mathbf{x}_i}{Dt} = \hat{\mathbf{u}}_i. \tag{3.41}$$

ϵ is a constant ($0 \leq \epsilon \leq 1$). With $\epsilon = 1$ the particles are moved with the field velocity, with $\epsilon = 0$ the particles are moved with their own velocity, normally $\epsilon = 0.5$. $\bar{\rho}_{ij} = (\rho_i + \rho_i)/2$. Details about the implementation are given in section 4.3. For consistency reasons, the velocity used to move the particles $\hat{\mathbf{u}}_i$ also has to be used in the conservation of mass equation. See equation 3.42 for the adapted conservation of mass equation. $\mathbf{u}$ is used in the Navier-Stokes equations.

3.4.4 Summary of equations in SPH

Now a summary of the main properties of the discretisation of the equations for fluid motion in SPH is given. Also the equations which form a closed set to simulate motion of fluid are summarized. The main properties are:

- Slightly artificial compressible fluid to get a time derivative of the density.
- With a stiff equation of state the pressure is calculated from the density.
- The pressure gradient in the Navier-Stokes equation is discretized in a symmetric way.
- Viscosity is modelled with an artificial viscosity approach.
- The particles are moved with an adjusted velocity according to the XSPH technique. The adjusted velocity is also used in the equation for conservation of mass.

The following closed set of equations describes fluid motion in a SPH model. Extra information about all used terms can be found at their original locations.

$$\frac{D\rho_i}{Dt} = \sum_j m_j (\hat{\mathbf{u}}_i - \hat{\mathbf{u}}_j) \cdot \nabla_i W_{ij} \tag{3.42}$$

$$p_i = B\left[\left(\frac{\rho_i}{\rho_0} \right)^7 - 1 \right] \tag{3.31}$$

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \mathbf{f}_i \tag{3.37}$$

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \epsilon \sum_j m_j \left(\frac{\mathbf{u}_j - \mathbf{u}_i}{\bar{\rho}_{ij}} \right) W_{ij} \tag{3.40}$$

$$\frac{D\mathbf{x}_i}{Dt} = \hat{\mathbf{u}}_i \tag{3.41}$$

Chapter 4

SPH implementation

In this chapter some implementations needed to make SPH suitable for simulations in hydraulic engineering are explained. When possible the influence of an implementation is visualised with a simulation of a breaking dam. The details about this simulation can be found in section 6.3. Here it is enough to know that it is about a square column of water with hydrostatic pressure placed behind a dam. At time $t = 0$ the dam is removed instantaneously and the fluid column collapses under influence of gravity. The flow configuration after $t = 0.14$ s will be used to show the influence of different implementations. The chapter ends with conclusions about the choices made for further simulations in this report.

4.1 Time integration

SPH is an explicit method and time integration can be done with standard numerical schemes like Runge-Kutta (RK), predictor corrector, Leap-Frog. For the SPH model in this thesis the second order Leap Frog is chosen because it has low memory usage and only one force evaluation per time step. The Courant-Friedrichs-Levy (CFL) condition has to be fulfilled. The CFL condition states that the distance fysical information or the particle itself can travel in one timestep has to be smaller than the numerical influence domain. The numerical influence domain is equal to the smoothing length h. Monaghan (1992) gave an expression for the stability condition in SPH:

$$\Delta t \leq \ \min\left(0.4\frac{h}{c + 0.6\alpha_{\Pi}c},\ \min\ 0.25(\frac{h}{f_i})^{1/2}\right), \tag{4.1}$$

where c is the used sound velocity, α is the used artificial viscosity parameter and f_i is the maximum magnitude of force per unit mass for a particle (acceleration).

In SPH there are only three independent variables: density ρ, velocity $\mathbf{u}$ and position $\mathbf{x}$. The pressure p is with the equation of state 3.31 directly linked to the density and is not an independent variable. As shown in section 3.4.4 there are three differential equations that

need to be integrated in time:

$$\frac{D\rho}{Dt} = f_1(\hat{\mathbf{u}}, \mathbf{x}), \tag{4.2}$$

$$\frac{D\mathbf{u}}{Dt} = f_2(\rho, \mathbf{u}, \mathbf{x}), \tag{4.3}$$

$$\frac{D\mathbf{x}}{Dt} = \hat{\mathbf{u}}. \tag{4.4}$$

$\hat{\mathbf{u}}$ is the velocity adjusted with XSPH (section 3.4.3) used to move the particle, $\mathbf{u}$ is the particle velocity from the momentum equations. Using the Leap-Frog time integration with predictors for velocity and density at intermediate time steps the differential equations are integrated in time as follows:

$$\rho^{n+1/2} = \rho^{n-1/2} + \Delta t f_1(\hat{\mathbf{u}}^{*n}, \mathbf{x}^n), \tag{4.5}$$

$$\mathbf{u}^{n+1/2} = \mathbf{u}^{n-1/2} + \Delta t f_2(\rho^{*n}, \mathbf{u}^{*n}, \mathbf{x}^n), \tag{4.6}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \hat{\mathbf{u}}^{n+1/2}. \tag{4.7}$$

How $\hat{\mathbf{u}}$ is integrated in time is explained in section 4.3. The predictors for density ρ^{*n} and velocity $\mathbf{u}^{*n}$ are defined by:

$$\rho^{*n} = \rho^{n-1/2} + 1/2\Delta t f_1(\hat{\mathbf{u}}^{*n-1}, \mathbf{x}^{n-1}), \tag{4.8}$$

$$\mathbf{u}^{*n} = \mathbf{u}^{n-1/2} + 1/2\Delta t f_2(\rho^{*n-1}, \mathbf{u}^{*n-1}, \mathbf{x}^{n-1}). \tag{4.9}$$

With these predictors the Leap-Frog time integration is second order accurate. In figure 4.1 the variables used in Leap-Frog are shown schematically in time.

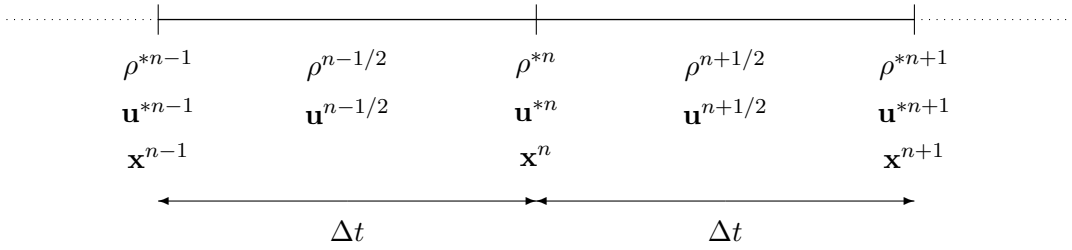


Figure 4.1: Variables in time for the Leap-Frog scheme.

4.2 Boundaries

4.2.1 Closed boundaries

When a fluid particle is coming close to a closed boundary, for example a wall, it is supposed to be blocked and it's fysical properties may not change. A water particle near a wall still has the density and other quantities of water. In a particle method this is not easy to implement, different choices are possible all with their strong and weak points. Three ways to make a closed boundary are mentioned here.

a) Boundary particles with repulsive forces

The closed boundary is modelled with a single row of boundary particles with mass, fixed density (1000 kg/m3), and no pressure. Their positions are fixed during the calculation. A repulsive boundary force is used to stop approaching fluid particles. When a fluid particle is approaching the repulsive force grows rapidly with decreasing distance. It works like the repulsive force between two molecules. Monaghan (Monaghan 1992) gave an expression of the boundary force per unit of mass:

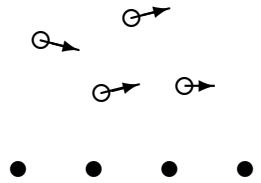


Figure 4.2: Fluid and boundary particles.

$$f(r_{ij}) = \begin{cases} D\left((\frac{r_0}{r_{ij}})^{12} - (\frac{r_0}{r_{ij}})^4\right)\frac{\mathbf{x}_{ij}}{r_{ij}^2} & r_{ij} \leq r_0 \\ 0 & r_{ij} > r_0 \end{cases} \tag{4.10}$$

$f(r_{ij})$ has dimension m^2/s, r_{ij} is the distance between the fluid and boundary particle. The length scale r_0 is chosen as the initial spacing between particles (Δr). The parameter D has dimension m^2/s^2 and is chosen of the same scale as gd, where d is a representative water level. The repulsive force works along the centreline of the fluid and boundary particle. It grows rapidly when the distance between fluid and boundary particle gets under r_0, and is zero when the distance is larger than r_0. To produce a slip boundary the boundary particles can simply be included in the calculation of viscous stress terms of fluid particles. The boundary particles are not used in the calculation of the averaged velocity with XSPH from section 3.4.3. The boundary particles are included in the density calculation of fluid particles near a boundary, but still these fluid particles are missing a contribution from other particles over the boundary. The boundary therefore disturbs the density of fluid particles nearby and also the pressure. Another drawback of this way of modelling a boundary is that the boundary force is radial around each boundary particle, it is not constant when moving along the boundary at a fixed distance. A flow is feeling this like ripples. The last drawback can be fixed by choosing another boundary force which is constant when moving along the boundary at a fixed distance. In fact this is already done (Monaghan et al. 2003), but it needs quite some computational effort.

B) Boundary particles with repulsive force and ghost particles

This way to model a closed boundary tries to minimize the disturbing influence of the boundary on the density and pressure of the fluid nearby. When a fluid particle is coming within the influence domain (κh) of a closed boundary, an identical ghost particle is mirrored at the other side of the boundary. This ghost particle has the same density and pressure as the original fluid particle but opposite velocity. The normal and tangential velocity at the boundary thus becomes zero. When a particle is coming closer to the boundary its mirrored ghost particle comes closer as well. The influence of the pressure of the ghost particle gets larger, but it is not enough to prevent penetration 100 %. These ghost particles are therefore combined with a single row of fixed boundary particles with a boundary force according to equation 4.10. Because of the combination of a repulsive force and ghost particles the constant D in equation 4.10 can be smaller than without ghost particles. When travelling parallel to the boundary particles are experiencing almost no ripples now. Fluid particles close to the boundary are not missing the contribution from outside the boundary because of the ghost particles. The density near the boundary is not disturbed very much. When a slip boundary is needed both ghost and boundary particles are used in the calculation of the viscous stress terms, for a free-slip boundary they are not used. Boundary and ghost particles are not used for the calculation of the XSPH term.

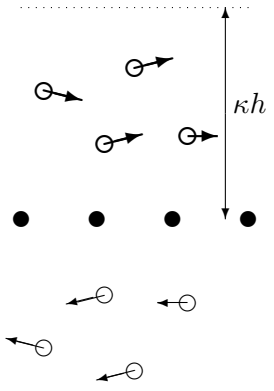


Figure 4.3: Fluid, boundary and ghost particles.

In summary: For every particle within κh from the boundary a ghost particle is mirrored outside the boundary every timestep. On the boundary there is a single line of boundary particles without pressure, but with a repulsive force for every fluid particle closer than r_{ij}. Both boundary and ghost particles are included in the density calculation of fluid particles. In plots of the results the ghost particles are not shown.

C) Quasi fluid boundary particles

Conceptual this way is the easiest one to model a closed boundary. A double row of fixed fluid particles is placed in a staggered way at the boundary. No estimates of boundary forces or directions are needed, the boundary particles just act like normal fluid particles except that their position is fixed in time. They build up pressure just like normal fluid particles, this is how they prevent particle penetration. Fluid particles near the boundary have interaction with a double staggered row of quasi-fluid particles and do not mis much interaction from outside. When a slip boundary is needed these quasi fluid particles are used in the calculation of the viscous stress terms, for a free-slip boundary they are not used.

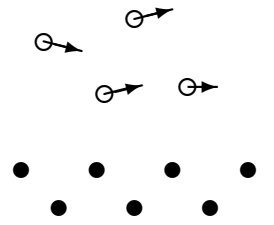


Figure 4.4: Fluid and quasi fluid particles.

Quasi fluid particles are not used for the calculation of the averaged velocity according to the XSPH technique.

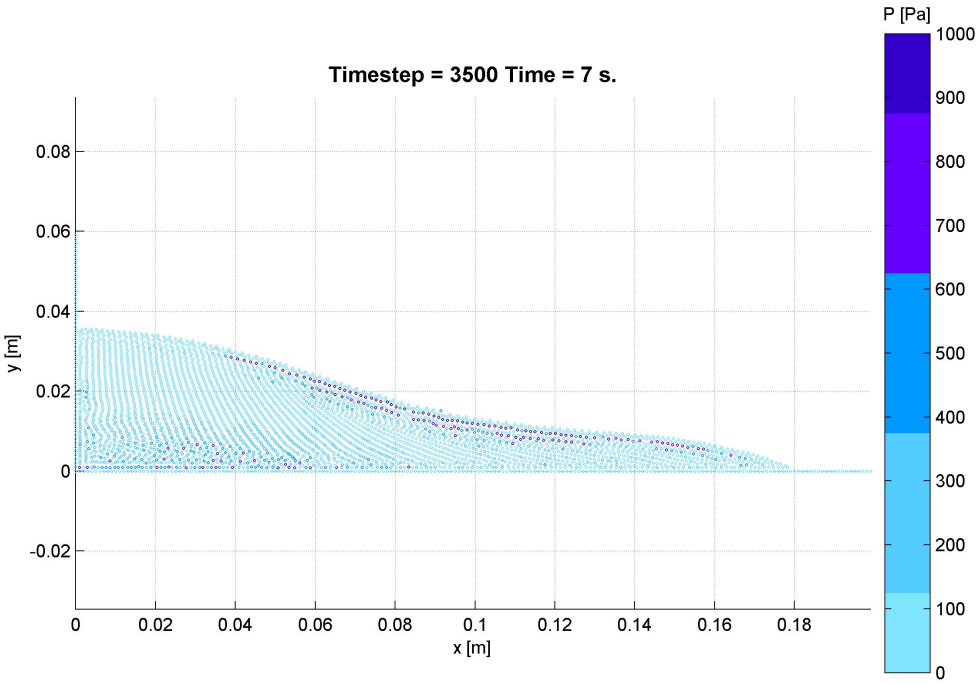
4.2.2 Influence of different closed boundaries

In this section the two most promising ways to model closed boundaries, boundary B and C, are compared with each other. Boundary particles combined with ghost particles, boundary B, will give less disturbance on the fluid than boundary particles only, which is boundary A. Therefore boundary A is not tested, and this section contains a comparison between boundary B and C.

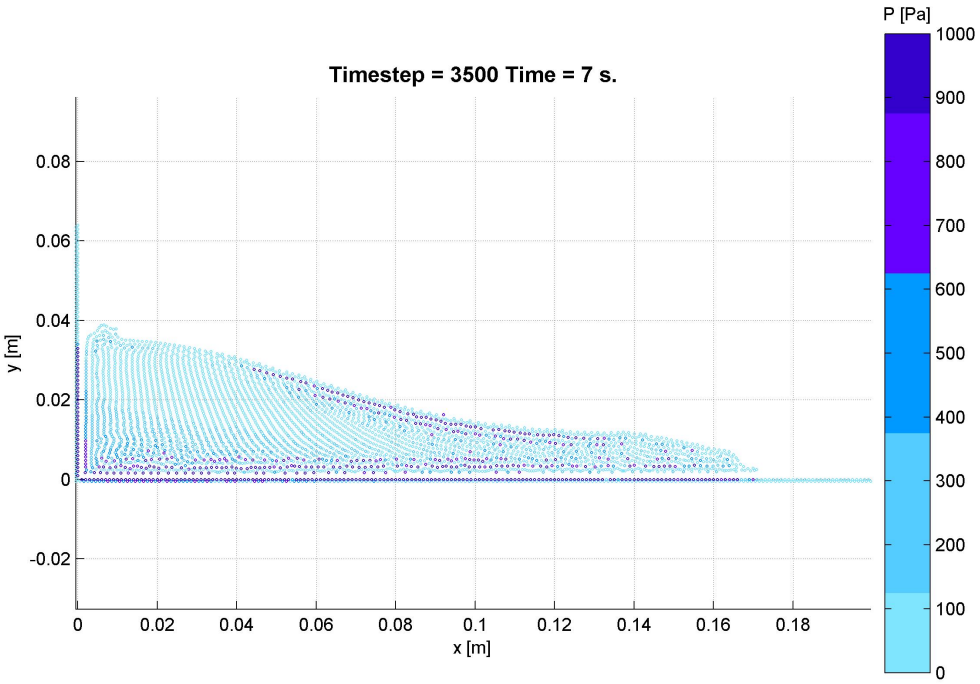
In figure 4.5 the result of the breaking dam simulation at time $t = 0.14$ s using boundary B and C are displayed. First, the simulation with boundary B is discussed. Be aware that only fluid and boundary particles are displayed, the ghost particles are not. The initial distance between the fluid and the boundary is equal to the initial distance between fluid particles Δr, this is also the influence distance of the boundary force. The pressure near the boundary is disturbed only a little by the boundary. The tip is pretty sharp, as expected with free-slip boundaries. The boundary has no negative influence on the shape of the water surface near the left boundary.

The result of the simulation with quasi fluid boundary particles in figure 4.5(b) is different. The results are sensitive to the initial distance between the fluid and this boundary. When the initial distance is too large the fluid can move towards the boundary and the initial condition is not at rest. When the initial distance is too low a fluid particle will feel suction when moving away from the boundary. This is not fysical, fluid moving away from a boundary is not glued to it. For the bottom the best initial distance between fluid and the boundary is $2\Delta r$, because this is the equilibrium distance between fluid and the bottom at places where there was no water initially (right of $x = 0.057$ m). The initial distance between the fluid and the left boundary is $2h = 2.8\Delta r$. A smaller initial distance gave suction and distorted the water surface. Now also the free surface is distorted left because the fluid can move towards the left boundary a little, but this distortion is smaller than with a smaller distance with suction. Free-slip boundaries are used, but the tip of the fluid is not smooth and sharp. It has a strange shape. This boundary also has a negative influence on the pressure of the fluid near the bottom.

With boundary B the particles can initially be placed at Δr from the boundary without the danger of giving under pressure when the particles are moving away. This together with less disturbing influence on the pressure near the wall does make this way of modelling a closed boundary preferable over boundary C. All further simulations are performed with closed boundaries consisting of boundary and ghost particles.



(a) Boundary 1 = boundary and ghost particles



(b) Boundary 2 = quasi fluid particles

Figure 4.5: Influence of two types of a closed boundary.

4.2.3 Open boundaries

At an open boundary the fluid can flow in or out the model. At an inflow boundary, particles have to be created and put in the system in a way that is not messing up the flow. At an outflow boundary, particles have to be removed without messing up the flow. At an open boundary only one parameter can be fixed, either the velocity/discharge, or the water level/pressure. But to put particles in the system in the correct way, one needs both position and velocity of each particle. Also the position and velocity of the top particle, which indicates the water level, need to be known exactly. The easiest way to satisfy these contradicting demands is to use a periodic boundary. With a periodic boundary the simulation is made round, particles which are leaving the domain on the right are entering with same fysical quantities on the left and vice versa. The particles within the influence distance of the left boundary are influenced by the particles within the influence domain of the right boundary and vice versa. The simulation behaves like the domain is truly round. A periodic boundary is a good way to model an open boundary without disturbing the fluid, but unfortunately it can only be applied to a limited number of cases.

Another combination of two open boundaries is used in this thesis for a 2D-vertical simulation. It is the simulation of water flowing over a sharp weir in section 6.7. At the inflow boundary on the left, the water level is horizontal and prescribed. No bottom friction is taken into account, so all particles in a cross section far away from the weir, move at roughly the same velocity. Within Δr from the inflow the horizontal inflow velocity is not prescribed, but the change in horizontal velocity is averaged to ensure that all particles keep on moving at the same velocity. A vertical line of particles therefore stays an exactly vertical line near the inflow. When a vertical line of particles has moved Δr away from the inflow, a new vertical line with exactly the same velocity is put into the domain. To prevent disturbance in the density and pressure of particles near the inflow boundary, the same approach as with closed boundaries is applied. Outside the inflow boundary two extra vertical rows with ghost particles at a horizontal distance of Δr are placed. These ghost particles make that the fluid particles near inflow are not missing interaction from their left. The outflow boundary is after the sharp weir. The water jet is just cut of, all particles lower than a prescribed position are removed. Because the particles are in the middle of a free fall the disturbing influence on the density is not important. No special treatment with ghost particles is made.

4.3 Moving the particles with XSPH

When using the XSPH technique from section 3.4.3 a particle is moved using an average between the velocity of the particle itself and the velocity of particles nearby. For consistency the velocity used to move the particles is the same as the velocity used in the conservation of

mass equations (equation 3.42). The equations describing the XSPH technique are:

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \epsilon \sum_j m_j \left(\frac{\mathbf{u}_j - \mathbf{u}_i}{\bar{\rho}_{ij}} \right) W_{ij}, \tag{3.40}$$

$$\frac{D\mathbf{x}_i}{Dt} = \hat{\mathbf{u}}_i. \tag{3.41}$$

ϵ is a constant ($0 \leq \epsilon \leq 1$) normally $\epsilon = 0.5$, $\bar{\rho}_{ij} = (\rho_i + \rho_i)/2$. This way of moving the particles conserves linear and angular momentum, but not energy. Monaghan (2005) describes a way to conserve energy with XSPH. Therefore this equation has to be made implicit by:

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \epsilon \sum_j m_j \left(\frac{\hat{\mathbf{u}}_j - \hat{\mathbf{u}}_i}{\bar{\rho}_{ij}} \right) W_{ij}, \tag{4.11}$$

now the averaged velocity is calculated using the already averaged velocities of the particles. An implicit formula like this is solvable by iteration, but that does take some iteration steps every timestep again for every particle. In this report the XSPH technique is used by taking the averaged velocities from half a timestep before to calculate the new averaged velocity. This prevents iteration. Using the same notation and time steps as in section 4.1 the averaged velocity now becomes:

$$\hat{\mathbf{u}}_i^{n+1/2} = \mathbf{u}_i^{n+1/2} + \epsilon \sum_j m_j \left(\frac{\hat{\mathbf{u}}_j^{*n} - \hat{\mathbf{u}}_i^{*n}}{\bar{\rho}_{ij}} \right) W_{ij}, \tag{4.12}$$

$$\hat{\mathbf{u}}^{*n} = \hat{\mathbf{u}}^{n-1/2} + 1/2 \Delta t f_2(\rho^{*n-1}, \mathbf{u}^{*n-1}, \mathbf{x}^{n-1}), \tag{4.13}$$

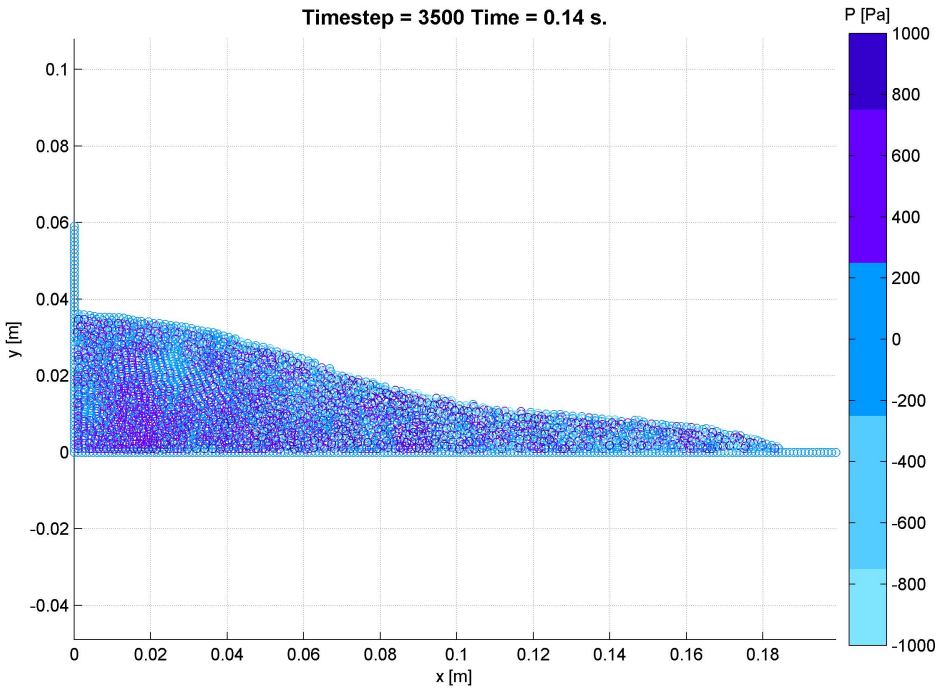
with subscripts i and j as particle indexes and superscripts indicating the timestep. Superscript * denotes a predicted value as explained in section 4.1. Function f_2 is the total time derivative of $\mathbf{u}_i$.

This way of calculating the averaged velocity is more energy conservative than the complete explicit equation 3.40, but without iteration. Note that every particle has two velocities. One indicates the amount of momentum of the particle, and is used to calculate the viscous stress terms. The other, averaged, velocity is used to move the particles and used in the conservation of mass equations.

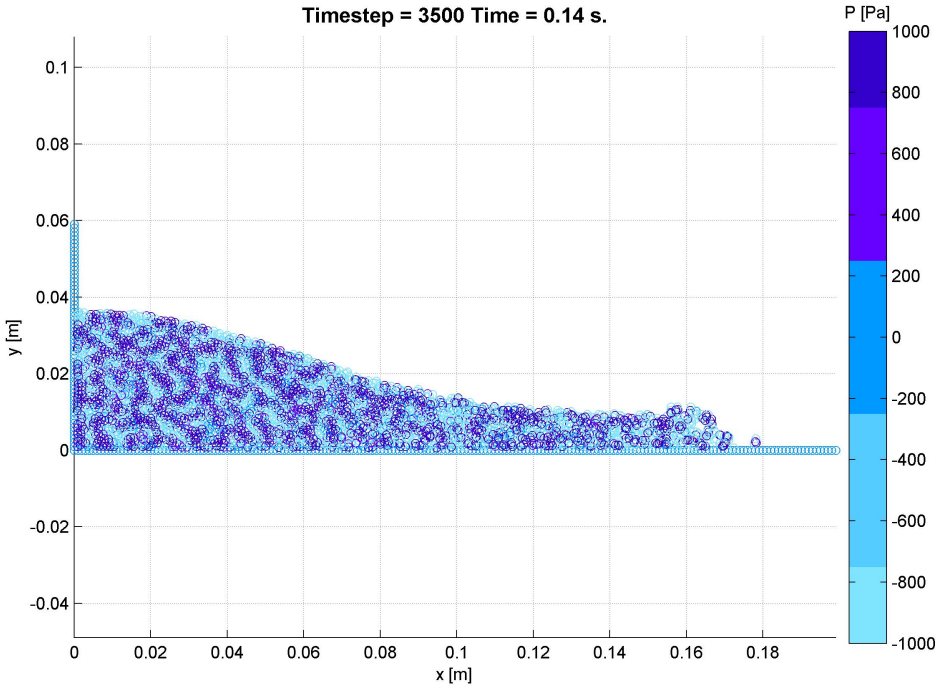
4.3.1 Influence of XSPH

To show the influence of the XSPH technique on the dam break problem, first the viscosity is lowered a lot. Originally the artificial viscosity was chosen as $\nu \approx \alpha hc/17 = 2.4 \cdot 10^{-4} \mathrm{m/s^2}$. In this simulation the viscosity parameter α is chosen as $\alpha = 0.01$, leading to $\nu \approx \alpha hc/17 = 0.9 \cdot 10^{-5} \mathrm{m/s^2}$. Viscosity decreases velocity differences by exchange of momentum. It makes the way particles move more orderly. XSPH also makes the way particles move more orderly by using an averaged velocity to move the particle. Their influence has some similarities, therefore the viscosity is lowered to make the influence of XSPH bigger.

In the comparison between moving the particles with XSPH or with just their own velocity in figure 4.6 the influence of XSPH is clear. With this low viscosity, clusters of particles with large negative pressure are next to clusters of particles with large positive pressure when XSPH is not used. At the front of the flow this even distorts the free surface. With XSPH the pressure field is smooth, without large areas of negative pressure and the free surface is still smooth and correct. XSPH does make the simulation more stable, prevents particle penetration without exchange of momentum and is an improvement.



(a) Move particles with XSPH.



(b) Move particles with their own velocity.

Figure 4.6: Influence XSPH.

4.4 Tensile instability

During simulations the particles in SPH can form clumps. This clustering is called tensile instability and is related to a combination of negative pressures and the sign of the second derivative. With the use of the Piecewise cubic spline the first derivative is negative and has its minimum at $q = r_{ij}/h = 2/3$, see figure 3.1. When particles are approaching each other because of negative pressure and their distance is already smaller than $q = 2/3$, the value of the first derivative becomes smaller and smaller and finally is zero. The repulsive force from the pressure gradient gets smaller in stead of bigger when two particles are too close and coming closer. This results in clumps of particles. The problem of tensile instability is tackled by Monaghan (2000) . An extra repulsive force between particles is introduced which is almost zero when $r_{ij} > \Delta r$ but increases a lot when $r_{ij} \to 0$. The adaptation to the standard SPH program are small. The original equation of conservation of momentum reads:

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \mathbf{f}_i. \tag{3.37}$$

Now an extra repulsive force term is introduced by replacing:

$$\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right)$$

with:

$$\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + R(f_{ij})^4 + \Pi_{ij} \right), \tag{4.14}$$

where the factor R depends on the pressure and density. The function f_{ij} is dependent on the kernel and the distance r_{ij} between the particles:

$$f_{ij} = \frac{W(r_{ij})}{W(\Delta r)}. \tag{4.15}$$

Δr is the initial particle spacing. With the Piecewise cubic spline and smoothing length $h = 1.4\Delta r$, $f_{ij} = 1$ when $r_{ij} = \Delta r$, $f_{ij} = 1.97$ when $r_{ij} = 0$ and $f_{ij} = 0.09$ when $r_{ij} = 2\Delta r$. This means that the repulsive force is growing a factor $(1.97)^4 = 15$ when the distance decreases from r_{ij} to zero. Also is the repulsive force practically zero ($(0.09)^4 \approx 0$) when the distance grows to $2r_{ij}$. The factor R is related to the pressure, and is defined by:

$$R = R_i + R_j, \tag{4.16}$$

with:

$$R_i = \begin{cases} \frac{0.2|p_i|}{\rho_i^2} & p_i < 0 \\ 0 & p_i \geq 0 \end{cases} \tag{4.17}$$

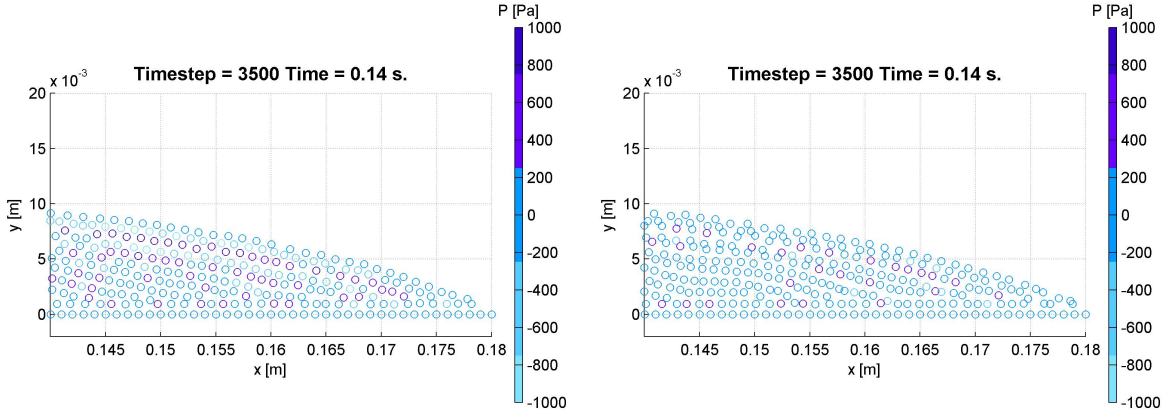
R_j can be found with replacing the index i with j. Even if both p_i and p_j are positive the particles tend to form local linear structures. This can be prevented by:

$$R = 0.01 \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \text{ if } p_i > 0 \wedge p_j > 0 \tag{4.18}$$

The influence of this last term is very small and mostly cosmetic. The repulsive force is much more important to prevent clumps when the pressure of a particle is negative. It can be seen as an artificial pressure term which is negligible when the distance between two particles is near Δr or larger and it comes in action when two particles are coming too close to each other.

4.4.1 Influence of artificial pressure

The influence of the artificial pressure term will be shown with a close up of the tip of the flow of the breaking dam simulation. The breaking dam simulation is executed as explained in the beginning of this chapter. One simulation shows the result with artificial pressure, the other is without artificial pressure. Both results are compared with each other in figure 4.7. When artificial pressure is used the particles are scattered in space more equally, and no clusters of particles are formed, but the pressure is varying more than in the simulation without artificial pressure. The artificial pressure does prevent particles coming too close, but it does not remove the cause for the particles to come that close to each other. The artificial pressure does prevent unfysical clustering of particles and will be used in further simulations.



(a) SPH with artificial pressure.

(b) SPH without artificial pressure.

Figure 4.7: Influence artificial pressure to prevent tensile instability.

4.5 Improvement of the accuracy of SPH

The accuracy of the SPH method can be improved by introducing corrections to the function approximations in SPH. Bonet e.a. (2000) wrote an article about corrections and stabilisation of SPH methods. In the article different corrections are proposed. The kernel itself can be adjusted, or the gradient. Also a combination of a simple correction to the kernel with a correction to the gradient is possible. The correction to the kernel itself is unsuitable for explicit type of calculations, because it is very computational expensive and the resulting equations are cumbersome to solve. This is the conclusion about kernel correction in another article addressing kernel normalisation (Bonet and Lok 1999). A simple gradient correction will be tried here. The correction to the SPH approximation of a gradient is proposed by Bonet e.a. (2004).

In SPH functions and there gradients are approximated with particle information as follows. See their original position in section 3.2.1 and 3.2.2 for explanation of all the terms.

$$\langle f(\mathbf{x}_i)\rangle = \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) W_{ij}, \tag{3.16}$$

$$\langle \nabla \cdot f(\mathbf{x}_i)\rangle = \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) \cdot \nabla_i W_{ij}. \tag{3.23}$$

Two requirements need to be fulfilled by the gradient approximation to ensure that the gradient of a linear or constant function is evaluated correctly. These two requirements are:

$$\sum_j \frac{m_j}{\rho_j} \nabla_i W_{ij} = 0 \text{ and } \sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) \otimes \nabla_i W_{ij} = \mathbf{I}. \tag{4.19}$$

A way to fulfill these two conditions is to correct the original gradient of the kernel $\nabla_i W_{ij}$ with a vector ϵ and a second order tensor $\mathbf{L}$. The corrected gradient of the kernel becomes:

$$\tilde{\nabla}_i W_{ij} = \mathbf{L}_i [\nabla_i W_{ij} + \epsilon_i \delta_{ij}], \tag{4.20}$$

with δ_{ij} as the Kronecker delta, and the correction terms ϵ_i and $\mathbf{L}_i$ defined by:

$$\epsilon_i = -\sum_j \frac{m_j}{\rho_j} \nabla_i W_{ij}, \tag{4.21}$$

$$\mathbf{L}_i = \left[\sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) \otimes \nabla_i W_{ij} \right]^{-1}. \tag{4.22}$$

These two correction terms on the gradient give a correct evaluation of constant and linear functions. All gradients are now approximated with the corrected gradient of the kernel $\tilde{\nabla}_i W_{ij}$:

$$\langle \nabla f(\mathbf{x}_i)\rangle = \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) \tilde{\nabla}_i W_{ij}. \tag{4.23}$$

4.5.1 Influence of gradient correction

This gradient correction has been implemented in the SPH code. For the correction term $\mathbf{L}_i$ the inverse of a matrix is needed. This inverse is very sensitive to the particle positions, near the fluid interface with only a couple of neighbours this matrix can become singular. Then the corrected gradient function is almost infinite, resulting in almost infinite acceleration of the particle. To reduce this problem a small positive term is added to the determinant of the matrix that has to be inverted. Two test cases are presented here. First is the breaking dam simulation, second is a still water test. For the second test a square of water particles with a bottom and two side boundaries consisting of boundary and ghost particles are subjected to gravity. Initially the particles do have zero pressure, in the end the simulation should give hydrostatic pressure.

Start with the breaking dam test. The parameters are chosen as explained in section 6.3. The result for SPH with gradient correction in figure 4.8 can be compared with the result without gradient correction in figure 4.5(a). With gradient correction a small clump of particles is ejected from the tip of the flow and shot to the right. Another particle is flying high above the rest of the flow. This is clearly unfysical and did not happen without the gradient correction. Near the free surface there is no line of particles with very high pressure as was in SPH without gradient correction, but the pressure near the bottom is worse. Even for this placid simulation the gradient correction caused unfysical ejection of particles from the free surface, that is not promising for more dynamic free surface problems.

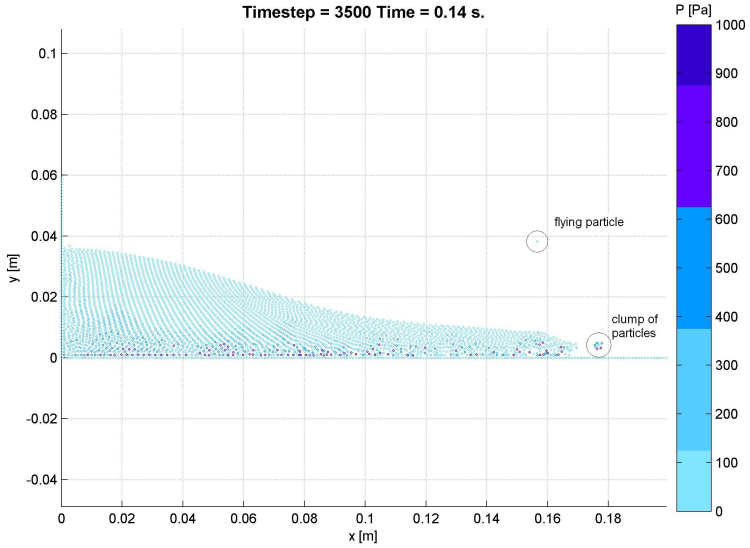
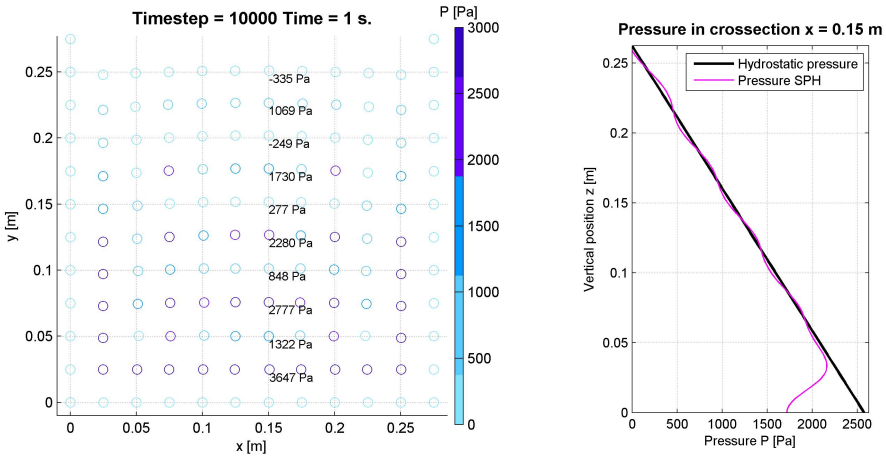


Figure 4.8: Breaking dam simulation SPH with gradient correction.

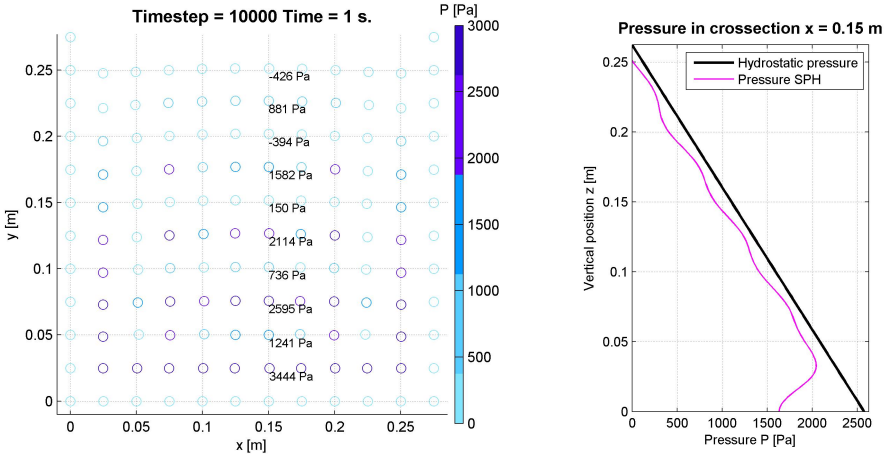
In the still water test the particles have initially zero pressure, but because of gravity the particles will feel the weight of the particles on top of them, this gives hydrostatic pressure. A large viscosity is used to damp out the velocities. As can be seen in figures 4.10 and 4.9, SPH does end up with pressure zero at the fluid interface and linearly increasing to the bottom. On the left the particles are displayed with the discrete values of the pressure of the particles in the middle. On the right a figure of the pressure in the vertical is displayed. The pressure is calculated at several points on a vertical line using the SPH approximation of a function, see equation 3.16. The pressure is varying largely from particle to particle.



(a) Particles with their pressure.

(b) Pressure in the middle.

Figure 4.9: Still water pressure SPH without gradient correction.



(a) Particles with their pressure.

(b) Pressure in the middle.

Figure 4.10: Still water pressure SPH with gradient correction.

But the oscillating behaviour of the pressure of discrete particles damps out mostly when the pressure is calculated on a vertical line using the SPH approximation of a function. Despite of the oscillating discrete values of the pressure at different particles, the pressure field still is close to the expected linear hydrostatic pressure. When the results of SPH with gradient correction are compared with the results of SPH without gradient correction some things can be concluded. First the gradient correction is not suppressing the pressure variation from particle to particle. Second the pressure at the vertical is correct for SPH without gradient correction, but too low with gradient correction.

This gradient correction gives disappointing results. It is meant to improve the evaluation of

linear and constant functions, but the pressure field is a linear function in the hydrostatic case and SPH with gradient correction analysed the pressure field worse than without correction. Also some particles get too much acceleration and are ejected from the rest of the flow. The correction does need several extra operations every timestep for every particle, and it makes the simulation about two times slower. Maybe some extra adaptations should be made to the SPH equations to get an improvement when using this correction technique, but in this thesis the gradient correction is not used in further simulations. Other corrections proposed by Bonet are more extensive as the gradient correction, but globally they use the same approach. Because of the bad results of the simple gradient correction the other more extensive corrections are not tried in this thesis.

4.6 Particle interaction

Because of the compact support of the smoothing function every particle has a finite number of neighbours which it has interaction with. When the smoothing length is chosen as 1.4 times the initial particle distance ($h = 1.4\Delta r$) every particle has 20 neighbours, see also figure 3.3. Because the position of all particles can change, the process of finding all neighbour particles within the influence domain is necessary every timestep. All interaction pairs are stored in an array. Starting with the first particle i the distance to all other particles $j = i+1, i+2, .., N$ is calculated and when this distance r_{ij} is smaller than the radius of the influence domain κh the indexes i and j are stored as an interaction pair. This is easy to program, but very time consuming to calculate. The total work to find all interaction pairs is of order $O(N^2)$ every timestep. There are more advanced ways to find all interaction pairs. A background mesh with a raster size equal to the influence distance κh can be used for book keeping. For each particle only 9 book keeping cells have to be considered in 2D where particles can be within the influence domain. This reduces the number of calculations of r_{ij} considerably, but for every problem a new background mesh has to be made. In the simulations for this report only the first way to find interaction pairs is used. Because of the conservation of linear momentum all interaction forces are symmetric. The influence of particle i on j is as big as the influence of j on i, but opposite in direction. Therefore interactions are not determined by calculating the influence of all neighbours on $i = 1, 2, .., N$ separately, but interaction is calculated per interaction pair.

4.7 Conclusions about chosen implementations

Here the findings of this chapter for the implementations in SPH will be summed up briefly. Time integration is performed with second order Leap-Frog. A closed boundary consists of ghost and boundary particles, open boundaries are rarely used and very problem dependent. Periodic boundaries are no problem in SPH. Particles are moved with an averaged velocity calculated with XSPH. Artificial pressure prevents unfysical clumping of particles. A gradient correction did not improve the accuracy of SPH, but made the results worse and is not used. All interaction pairs are found by just checking all possible pairs.

Chapter 5

2D SPH computer code

The equations from chapter 3 have been implemented in a 2 dimensional computer code using chapter 4 for further specifications. The computer code explained in the book SPH (Liu and Liu 2003) has been a help when developing the code used in this thesis, but the program explained here is different and programmed by the author from scratch. Two basic versions of the program are explained, the first version has fixed closed boundaries and no open boundaries, it is called 'SPH 2D closed'. The second version can have moving boundaries and inflow or outflow of particles, this version is called 'SPH 2D open'. In section 5.1 the global outline of both versions is explained with their differences. All used files for both versions are explained in section 5.2 and 5.3. How SPH 2D can be used is explained in section 5.4.

5.1 Program outline

In order to model the motion of fluid with SPH the equations of chapter 3 are solved together with the implementations from chapter 4. Next to the main properties of the solved equations in SPH from section 3.4.4 the following features are important for the implementation of 'SPH 2D closed' in a computer code.

- SPH 2D programmed in Fortran 77
- Input of fluid and boundary particles from file or generated by the program initially
- Ghost particles are generated at prescribed boundaries
- Interactions are found by simply checking all possible pairs
- The Piecewise cubic spline kernel is used for function approximations
- Artificial viscosity, XSPH, or artificial pressure can be used if wanted
- Time integration with explicit second order Leap Frog

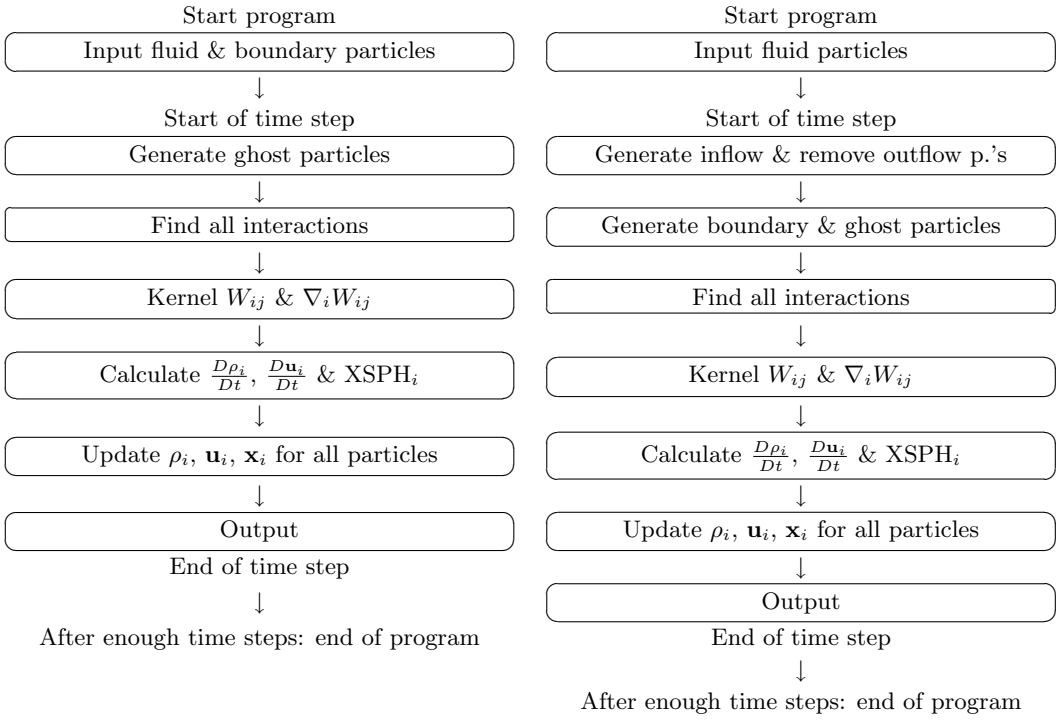


Figure 5.1: 'SPH 2D closed' layout (left) 'SPH 2D open' layout (right).

Because of the explicit time integration, the CFL condition need to be satisfied, so the permissible time steps are of the order $O(10^{-4}s)$ for a commonly used particle spacing $\Delta r = 0.025$ m. Next to the features of the closed version, the open version can have moving boundaries and open boundaries. This results in the following two extra features.

- Boundary particles are generated every time step, not read from inputfile
- At inflow particles are generated and at outflow removed

The global outline of the two versions of the program are shown next to each other in figure 5.1. In the open version every time step the subroutine for inflow and outflow is called, here particles are generated or removed if needed. Because with open boundaries the total number of particles can change every time step and the boundary particles are stored in the same array as the fluid particles, the boundary particles have to be generated every time step whether the boundary is moving or not. In the closed version the boundary particles have to be generated only once, this operation can be put in front of the time loop. So the closed version is a little faster, because less operations are necessary every time step, but the open version is more flexible.

5.2 SPH 2D closed files

The used Fortran files for 'SPH 2D closed' are briefly explained. The files for this version are not listed in the appendix, but the more extensive files for 'SPH 2D open' can be found in

appendix A. In the files it is clearly marked which parts of the code are not used in 'SPH 2D closed'. The short description per file including all the calls to each other will make clear how the program is implemented in Fortran code.

`SPH_2D.f`	Main program. Calls `input.f` once, starts time loop with `derivatives.f` and Leap Frog time integration every time step. Calls `output.f` if requested, generates output on screen.
`input.f`	Subroutine for initial input. Generates or reads from file all fysical quantities for fluid particles and boundary particles.
`derivatives.f`	Subroutine to solve the equations every time step. Make all time derivatives zero at start. Generation of ghost particles. Interaction-pairs, kernel W_{ij} & $\nabla_i W_{ij}$, and $\frac{D\rho_i}{Dt}$ are found. Get pressure from density. Pressure terms including artificial pressure, artificial viscous terms, gravity and boundary forces determined. $\frac{D\mathbf{u}_i}{Dt}$ found. Influence of average velocity calculated with XSPH. First time step call `parameterfile.f`.
`output.f`	Subroutine to write output to textfile. Writes output in textfile with 8 columns containing particle information i, x, y, u_x, u_y, ρ, p, m.
`parameterfile.f`	Subroutine to write general info about simulation in textfile.
`param.txt`	Include file with some declarations and parameters. This file is included in every other file. It contains the declaration of some variables used in every file. Contains parameters telling if the input is from file or has to be generated by the program. And contains parameters telling if artificial viscosity, XSPH or artificial pressure is used.

5.3 SPH 2D open files

The same Fortran files as for 'SPH 2D closed' plus two extra files `inputpb.f` and `inoutflow.f` are used for 'SPH 2D open'. `derivatives.f` and `input.f` are slightly different, which is clear from the description below. The complete listing of 'SPH 2D open' can be found in appendix A. In the complete code it is clearly marked which parts are only used in 'SPH 2D open', all other parts are used in both versions. The displayed code is used for the flow over a sharp weir simulation in section 6.7, some parts are specific for that simulation, which also is indicated clearly. The following will make clear how 'SPH 2D open' is implemented in Fortran.

`SPH_2D.f`	Main program. Calls `input.f` once, starts time loop with `derivatives.f` and Leap Frog time integration every time step. Calls `output.f` if requested, generates output on screen.
`input.f`	Subroutine for initial input. Generates or reads from file all fysical quantities for fluid particles only.
`derivatives.f`	Subroutine to solve the equations every time step. Generation of ghost particles. Calls `inoutflow.f` and `inputbp.f`. Now make all time derivatives zero. Interaction pairs, kernel W_{ij} & $\nabla_i W_{ij}$, and $\frac{D\rho_i}{Dt}$ are found. Get pressure from density. Pressure terms including artificial pressure, artificial viscous terms, gravity and boundary forces determined. $\frac{D\mathbf{u}_i}{Dt}$ found. Influence of average velocity calculated with XSPH. First time step call `parameterfile.f`.
`inoutflow.f`	Subroutine for inflow and outflow. Inflow particles are generated and added to the existing arrays. Outflow particles are removed from the existing arrays.
`inputbp.f`	Subroutine for generation of boundary particles. Generates all boundary particles. In 'SPH 2D closed' this part is included in `input.f`.
`output.f`	Subroutine to write output to textfile. Writes output in textfile with 8 columns containing particle number i, x, y, u_x, u_y, ρ, p, m.
`parameterfile.f`	Subroutine to write general info about simulation in textfile.
`param.txt`	Include file with some declarations and parameters. This file is included in every other file. It contains the declaration of some variables used in every file. Contains parameters telling if the input is from file or has to be generated by the program. And contains parameters telling if artificial viscosity, XSPH or artificial pressure is used.

5.4 How to use SPH 2D

SPH 2D does not have an interface, all input has to be specified in the Fortran files. The Fortran files can be compiled with any Fortran 77 compiler into an executable. SPH 2D does not have a consistency check on the input, all input must be correct, otherwise the result of the simulation will be nonsense. It will be explained here which variables can be adjusted where in the program, but first think about the problem that has to be simulated with SPH 2D.

- What is the problem domain, where are the boundaries?
- Is the problem 2Dv or 2Dh: gravity needed or not?
- What is the desired resolution, this determines the initial particle-spacing Δr.

- What will be the maximum velocity in the simulation, $c \approx 10u_{max}$?
- What time step is required to keep the simulation stable, how many time steps?
- What are the initial conditions, generate them with the program or read them from files?
- Use artificial pressure, artificial viscosity or average velocity XSPH?
- What viscosity parameter α and what XSPH ϵ is needed?

These questions are mostly obvious, but all of them need an answer. File by file the important parameters that have to be defined for every simulation will be explained.

`param.txt`	Parameters for the use of artificial pressure, artificial viscosity, average velocity XSPH and input from file or not can be changed here.
`input.f`	Input from file → give source-directory. Generating input → give initial particle parameters.
`inputbp.f`	Program position of the boundary particles. Other fysical parameters must be consistent with the fluid particles. *** In 'SPH 2D closed' this part is included in `input.f` ***
`inoutflow.f`	Inflow → program the position and velocity of inflow particles and let them flow into the domain at the right time. Other fysical parameters must be consistent with the already existing fluid. Outflow → remove all parameters for outflowing particles from the arrays. *** Not in 'SPH 2D closed' ***
`derivatives.f`	Give the position of closed boundaries for generation of ghost particles. Choose $c \approx 10u_{max}$, c comes back in $B = c_0^2\rho_0/7$ for the equation of state 3.31. Set α for artificial viscosity and ϵ for average velocity XSPH. Set influence domain of boundary force r_o. Normally $r_o = \Delta r$. Also set constant D for the boundary force. Comment the lines for the influence of gravity when gravity is not needed.
`SPH_2D.f`	Set the time step, maximum number of time steps and the desired frequency of output here.

When all these parameters are chosen correctly the program will produce one textfile every chosen time step containing 8 columns with the information of all the fluid and boundary particles (i, x, y, u_x, u_y, ρ, p, m). Matlab can be used to post-process the textfiles.

Chapter 6

Simulations

With the computer code described in chapter 5 simulations in the field of hydraulic engineering are carried out. Section 6.2 deals with some viscosity benchmark problems to test the formulation of viscous shear stresses in SPH. In the rest of this chapter, problems with a free surface are treated. These is a simulation of a breaking dam, a bore at a wall, a standing wave in a closed basin, waves propagating on a beach, and finally a sharp weir. These simulations will show the accuracy of SPH. But first the interpretation of results in SPH is explained in next section.

6.1 Interpretation of results

The interpretation of results from SPH is not always straightforward, here some important notions are made. In SPH space is discretised using particles. Every particle has the quantities mass, density, pressure, velocity, and position. Other quantities like color, or salinity etc. could be stored for every particle, but that has not been done in this thesis. The result of a simulation with SPH is a data file for every required timestep with all particle quantities stored. These quantities are addressed to the centre of each particle. In 2D every particle represents a certain area $(\Delta r)^2$ around its centre. In figure 6.1 the space discretisation with particles is shown, ghost particles are not shown. Near boundaries one has to be careful when interpreting the results. As can be seen the transition between fluid and the wall is not at the centreline of the boundary particles, but somewhere between the closest fluid particle and a boundary particle. Every fluid particle represents an area of $(\Delta r)^2$, a correct definition of the boundary seems to be $1/2\Delta r$ under the lowest fluid particles and $1/2\Delta r$ next to the most left or right fluid particles.

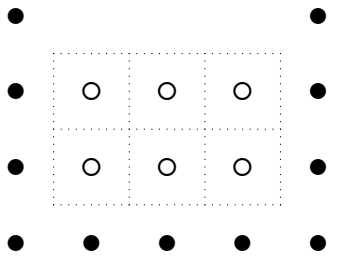


Figure 6.1: SPH space discretisation with boundary and fluid particles. Dotted line shows the area of every fluid particle.

With Matlab a snapshot of the position of the particles can be made to visualize the flow

from the data files. Sometimes information is needed on a specific point of interest. There are not always particles exactly at the point of interest, therefore the function approximation in SPH is used:

$$\langle f(\mathbf{x}_i)\rangle = \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) W_{ij}, \tag{3.16}$$

now $\mathbf{x}_i$ is the position of a point of interest. For example, to calculate the velocity at a point of interest this equation becomes:

$$\langle \mathbf{u}_i\rangle = \sum_j \frac{m_j}{\rho_j} \mathbf{u}_j W_{ij}. \tag{6.1}$$

$\langle \mathbf{u}_i\rangle$ is the velocity at a point of interest found by the average of the velocity of particles nearby. For the flow parameters in a cross-section several points of interest are chosen at a vertical line. When the density is calculated at a vertical line, the position where the density gets under 500 kg/m^3 is defined as the free surface. This definition of the free surface roughly gives a free surface at $1/2\Delta r$ above the centre of the highest particles. This agrees with the area $(\Delta r)^2$ every particle represents in figure 6.1. From the data files the flow configuration, information at a certain point of interest, and the water depth can be found. This is all the information needed to analyse the simulations.

6.2 Viscosity benchmark problems

To test how effective the artificial viscosity is in practice, three standard benchmark problems are used. These are a Poisseuille flow, a Couette flow and a shear driven cavity simulation, all with very low Reynolds numbers. The Reynolds number is given by $Re = ul/\nu$. u is a characteristic velocity, l is a characteristic length scale and ν is the kinematic viscosity. A low Reynolds number means the flow is laminar and not turbulent.

6.2.1 Poisseuille flow

The first viscosity test involves the flow between two parallel infinite plates placed at $y = 0$ and $y = l$. Initially all velocities are zero, but because of a constant driving force F the fluid will start to flow to the right. With a constant kinematic viscosity the final state is a stationary parabolic velocity profile with zero velocity at the wall because of drag and its maximum in the middle. The velocity profile can be found from the Navier-Stokes equations. Velocities in y direction are zero, only the Navier-Stokes equation in the horizontal x need to be considered. In the stationary final state the time derivative of the velocity is zero. There is no pressure gradient and the velocity gradient in horizontal x direction is zero. The only body force is the constant driving force F in the horizontal x direction, this results in:

$$0 = \nu \frac{\partial^2 u_x}{\partial y^2} + F. \tag{6.2}$$

Using that $u_x = 0$ at both boundaries $y = 0$ and $y = l$ this equation can be integrated to y twice, giving the final state solution of the velocity profile:

$$u_x(y) = -\frac{F}{2\nu} y(y - l). \tag{6.3}$$

The maximum velocity in the final state occurs at $y = 1/2l$ and is given by:

$$u_{x,max} = -\frac{F}{8\nu} l^2. \tag{6.4}$$

The development of this solution in time from the start to the final state is found in the book of Liu (2003):

$$u_x(y,t) = -\frac{F}{2\nu} y(y - l) + \sum_{n=0}^{\infty} \frac{-4Fl^2}{\nu\pi^3(2n+1)^3} \sin(\frac{\pi y}{l}(2n+1)) \exp(-\frac{(2n+1)^2\pi^2\nu}{l^2} t). \tag{6.5}$$

In this simulation the length $l = 10^{-3}$ m, the kinematic viscosity $\nu = 10^{-6}$ m^2/s, and the driving force $F = 2 \cdot 10^{-5}$ m/s. Using equation 6.4 this results in a maximum velocity $u_x = 2.5 \cdot 10^{-5}$ m/s, which corresponds to $Re = u_x l/\nu = 0.025$.

In SPH the infinite length of the plates is simulated with periodic boundaries in the flow directions. Periodic boundaries are explained in section 4.2.3. The used parameters in the SPH simulation together with the initial particle setup are showed below.

Poisseuille flow simulation:
20x39 = 780 particles, $\Delta r = 2.5 \cdot 10^{-5}$ m
Smoothing length $h = 3.5 \cdot 10^{-5}$ m
$c = 0.25$ m/s
$\Delta t = 0.00005$ s
Artificial viscosity
$\alpha = 1.95$, $\nu \approx \alpha hc/17 = 1 \cdot 10^{-6}$ m/s^2
Slip boundaries up and down
$D = 0.01$ m^2/s
Periodic boundaries left and right
No gravity

Initial conditions:
$u_x = 0$, $u_y = 0$

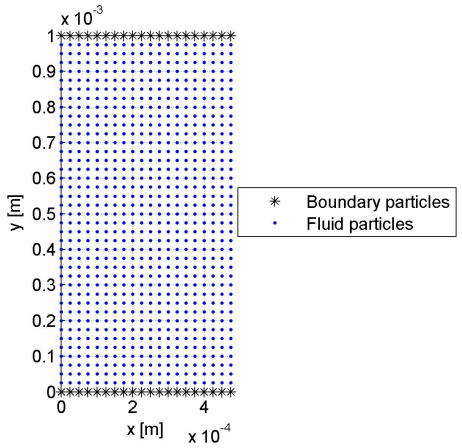


Figure 6.2: Initial particle positions Poisseuille flow.

After 12000 steps ($t = 0.6$ s) the simulation reaches the steady end state. The particles with some velocities at $t = 0.6$ s are plotted in figure 6.3. Only fluid and boundary particles are shown, ghost particles at the top and bottom boundary or left and right are not shown. The velocity profiles at different times from SPH are compared with the analytical solution according to equation 6.5 in figure 6.4. The agreement is excellent. A conclusions that can be drawn from this benchmark test is that although artificial viscosity doesn't use second derivatives of the velocity, it can predict a parabolic velocity profile without problems. The kinematic viscosity resulting from these results is $\nu \approx \alpha hc/17$, this is slightly less than the theoretically derived $\nu \approx \alpha hc/16$ in section 3.4.2. The same conclusion can be drawn from tests where h, c, and α are changed in such way that the kinematic viscosity $\nu = 1 \cdot 10^{-6}$ m/s^2 stays the same. Also in simulations with a different ν the velocity profiles from SPH agree with the theoretic profiles when the kinematic viscosity is calculated using $\nu \approx \alpha hc/17$.

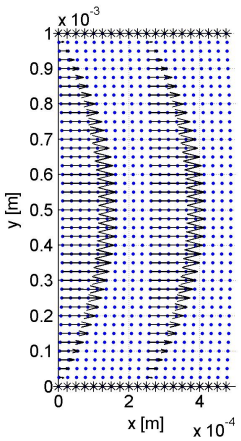


Figure 6.3: Poiseuille flow at $t = 0.6$ s.

During this laminar and ordered simulation a spurious checkerboard mode enters the pressure. The reason for this is that all variables are known at every particle and in this ordered situation the distance between all neighbours is constant. The contribution of a neighbour on a particle is the same as the contribution of the opposite neighbour. A checkerboard mode in pressure now gives zero pressure gradient at every particle, this is an artificial equilibrium. SPH is normally used in very dynamic situations, then the distance between particle is not the same for all neighbours and this checkerboard mode does not show up. At $t = 0.6$ s the checkerboard mode is very weak, but not totally absent, it slows down the flow a little. The influence is less than one percent. This influence can be seen from the fact that SPH is overpredicting the velocities a little at $t = 0.1$ s when the checkerboard mode is not developed yet, but SPH is exactly predicting the velocities at $t = 0.6$ s with the checkerboard mode.

6.2.2 Couette flow

The second viscosity test is almost the same as the first and also involves the flow between two parallel infinite plates placed at $y = 0$ and $y = l$. Initially all velocities are zero, but suddenly at $t = 0$ the upper plate starts to move to the right with U_0. Again the stationary velocity profile in the final state can be found from the Navier-Stokes equations. Velocities in y direction are zero, only the Navier-Stokes equation in the horizontal x need to be considered. In the stationary final state the time derivative at the LHS is zero, there is no pressure gradient, the velocity gradient is zero, and there is no body force. This results in:

$$0 = \frac{\partial^2 u_x}{\partial y^2}. \tag{6.6}$$

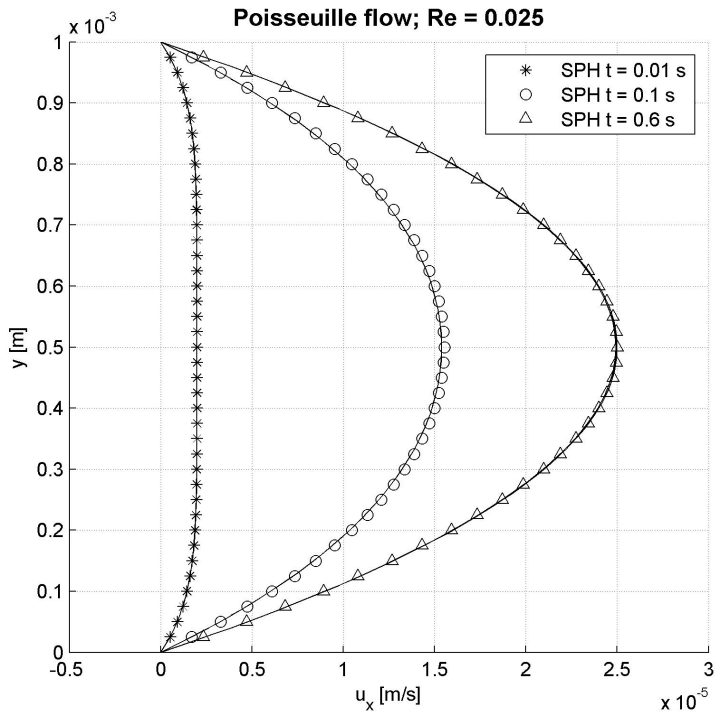


Figure 6.4: Comparison Poisseuille flow results SPH (marks) with theory (line).

Using that $u_x = 0$ at the under plate $y = 0$ and $u_x = U_0$ at the upper plate $y = l$ this equation can be integrated to y two times leading to the final state solution of the velocity profile:

$$u_x(y) = \frac{U_0}{l} y. \tag{6.7}$$

This is a linear velocity profile with zero velocity at $y = 0$ and maximum velocity of $u_x = U_0$ at $y = l$. The development of this solution in time from the start to the final state is found in the book of Liu (2003):

$$u_x(y,t) = \frac{U_0}{l} y + \sum_{n=1}^{\infty} \frac{2U_0}{n\pi} (-1)^n \sin(\frac{n\pi}{l} y) \exp(-\frac{n^2\pi^2\nu}{l^2} t). \tag{6.8}$$

In this simulation the length $l = 10^{-3}$ m, the kinematic viscosity $\nu = 10^{-6}$ m^2/s, and $U_0 = 2.5 \cdot 10^{-5}$ m/s. The Reynolds number for this test is $Re = u_x l/\nu = 0.025$.

The geometry, initial particle positions and chosen parameters in the SPH calculation are the same as with Poisseuille flow and can be found there. Here only the result of SPH will be compared with the analytical solution in figure 6.5. At $t = 0.6$ s the stationary state is reached. As with Poisseuille flow both the development in time as the final state solution from SPH agrees excellent with the analytical solution. The final state is the theoretic derived linear velocity profile. Here also the kinematic viscosity is calculated with $\nu \approx \alpha hc/17$.

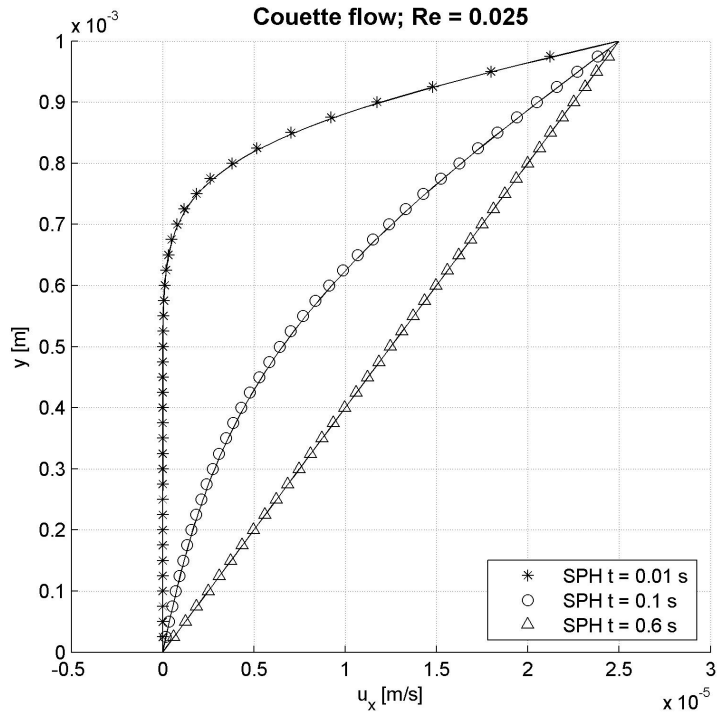


Figure 6.5: Comparison Couette flow results SPH (marks) with theory (line).

6.2.3 Shear driven cavity

The last viscosity test is a shear driven cavity problem. A square closed box of fluid is considered with four closed boundaries. The upper boundary moves with U_0 to the right, while the other boundaries have a zero velocity. The flow will reach a steady circulation pattern of which the shape is determined by the Reynolds number. In this simulation $U_0 = 0.001$ m/s, the length of a side of the square box $l = 0.001$ m and the kinematic viscosity $\nu = 1 \cdot 10^{-6}$ m/s^2. $Re = U_0 l/\nu = 1$. This is a 2D problem and a theoretic solution cannot be simply derived from the Navier-Stokes equations. The results of SPH are compared with results of a finite volume method (FVM) from Wesseling (2002). In the FVM calculation 40 x 40 cells are used. The parameters in SPH together with the initial particle set up are showed below.

Shear driven cavity simulation:
39x39 = 1521 particles
$\Delta r = 2.5 \cdot 10^{-5}$ m
Smoothing length $h = 3.5 \cdot 10^{-5}$ m
$c = 0.25$ m/s
$\Delta t = 0.00005$ s
Artificial viscosity $\alpha = 1.95$
$\nu \approx \alpha hc/17 = 1 \cdot 10^{-6}$ m/s^2
Slip boundaries
$D = 0.01$ m^2/s
No gravity

Initial conditions:
$u_x = 0$, $u_y = 0$

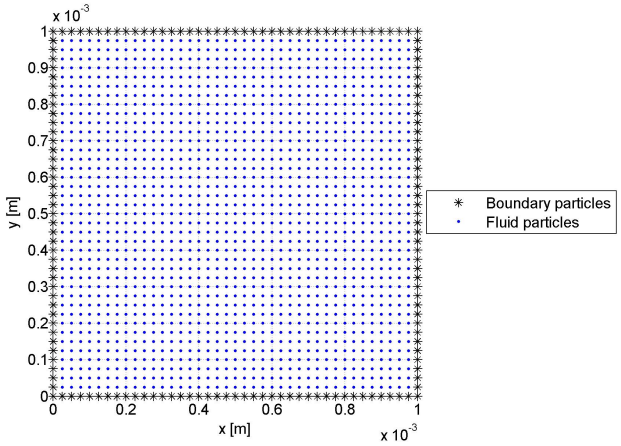


Figure 6.6: Initial particle positions shear driven cavity.

In figure 6.7 the situation at $t = 0.15$ is plotted. The steady state is reached now. A circulation pattern has been developed with its centre at $x \approx 1/2l$ and $y \approx 0.78l$. Because the Reynolds number is very small there is only one major circulation, there are no secondary circulations in the corners. SPH gives a correct circulation pattern, except very close to both upper corners. The velocity of the particles in both corners is too low, maybe the boundary forces around the boundary particles slows down the flow too much. The non-dimensional velocity profiles over the horizontal and vertical centerlines from SPH are compared with the velocity profiles from FVM in figure 6.8. The shape of the velocity profile in SPH is correct, but it underestimates the maximums with 19 %.

6.2.4 Conclusion about viscosity benchmark problems

Three benchmark problems have been used to test the accuracy of the artificial viscosity approach in SPH. Artificial viscosity actually decreases velocity differences between particles by exchange of momentum and it simulates viscous stresses without the use of second derivatives of velocities. The Poisseuille flow and Couette flow showed that the artificial viscosity

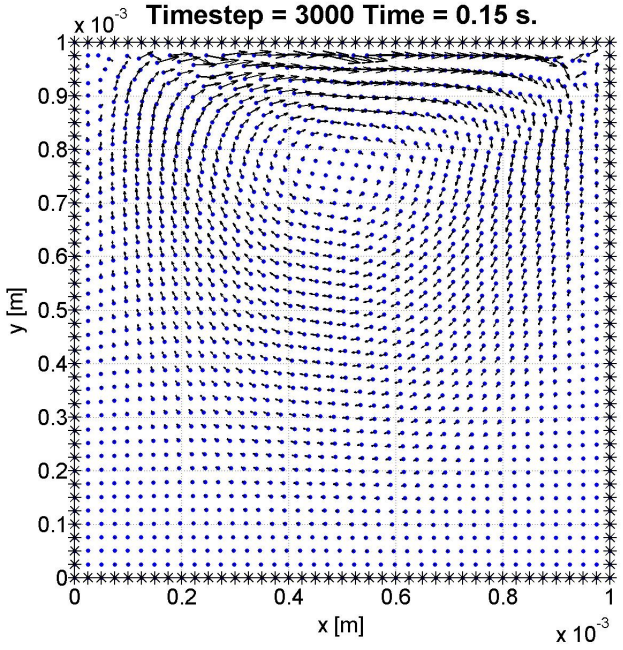
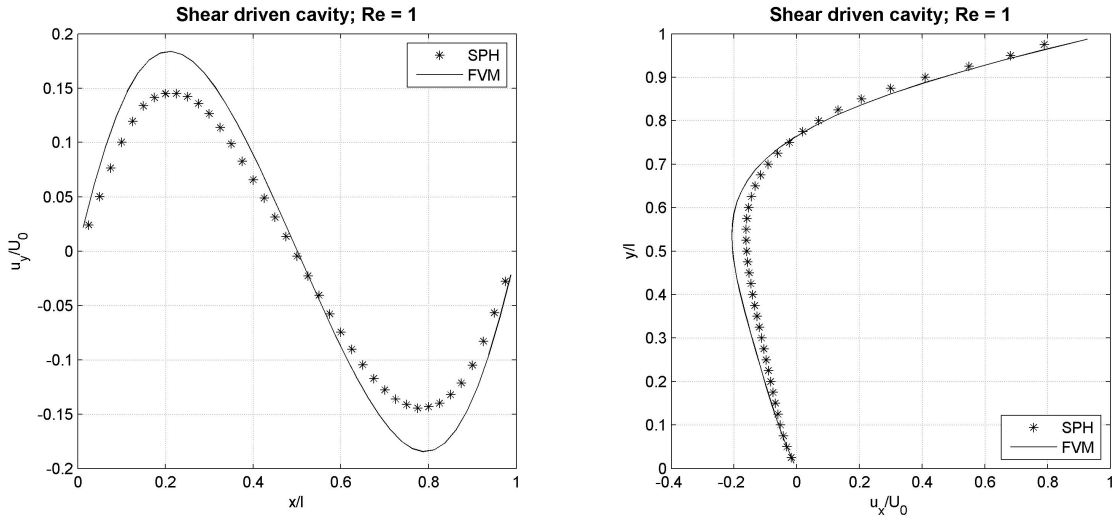


Figure 6.7: Shear driven cavity at $t = 0.15$ s.



(a) Normalised velocity at horizontal centerline

(b) Normalised velocity at vertical centerline

Figure 6.8: Comparison shear driven cavity results SPH with FVM at $t = 0.15$ s.

gives an effective kinematic viscosity parameter $\nu \approx \alpha hc/17$, this is slightly smaller than the theoretic expected $\nu \approx \alpha hc/16$. The development in time and the final state velocity profile is exactly as expected. In the shear driven cavity test SPH ended in a steady circulation pattern as expected, but SPH under predicted the maximum velocities with 19 %. The three tests showed that the artificial viscosity approach can give a linear velocity profile for Couette flow, a parabolic velocity profile for Poiseuille flow, but a sharper curved velocity profile in a

shear drive cavity was more difficult. With these conclusions in mind now simulations with a free surface will be carried out.

6.3 Breaking dam

The breaking dam simulation describes the flow of water after a dam is broken. SPH is very suitable to describe this problem with a fast varying water level. Initially a square column of water with hydrostatic pressure is placed behind a dam. At time $t = 0$ the dam is removed instantaneously and the fluid column collapses because of gravity. In the past similar simulations have been executed with the Marker-and-Cell method (Harlow and Welch 1965), and several times with SPH, (Monaghan 1992) and (Bonet and Lok 1999). The results of the simulation will be compared with experimental data (Martin and Moyce 1952). In the experiment the water column is 0.05715 m (2.25 inch) high and 0.05715 m wide, it is square. In the SPH calculation this is approximated with 57 by 57 particles with $\Delta r = 0.001$. The water column in SPH is therefore 0.057 m by 0.057 m. Free slip boundaries are used with a weak boundary force. The most important parameters together with the initial particle positions are mentioned below.

Breaking dam simulation:
57x57 = 3249 particles, $\Delta r = 0.001$ m
Smoothing length $h = 0.0014$ m
$c \approx 10\sqrt{2gd} = 10.59$ m/s
$\Delta t = 0.00004$ s
Artificial viscosity
α=0.27, $\nu \approx \alpha hc/17 = 2.4 \cdot 10^{-4}$ m/s^2
Free slip boundaries
$D = 0.1$ m^2/s

Initial conditions:
$u_x = 0$, $u_y = 0$
Hydrostatic pressure

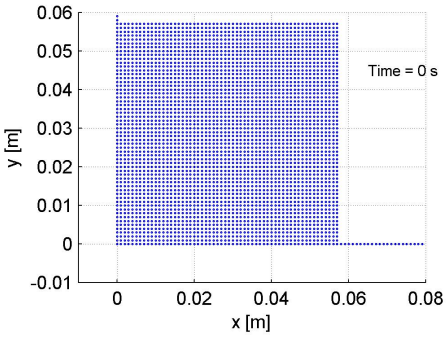


Figure 6.9: Initial particle positions breaking dam.

The position of the fluid at six different times is displayed in figure 6.10. First the fluid is squeezed out at the bottom of the column, later the top of the column moves down. The tip of the fluid is sharp, as expected with free slip boundaries. The surface stays smooth during the whole simulation. The shape of the fluid during the simulation is comparable with the shape using the Marker-and-Cell method or earlier simulations with SPH, it is also comparable with the shape in the experiment (Martin and Moyce 1952). Some small differences in the shape of the surface near the boundaries are visible though. SPH uses an artificial viscosity model with constant ν, therefore wall friction would overestimate the drag and free slip boundaries are used. Only the radial boundary force around every boundary particle gives ripples and slows down the flow a little bit near boundaries in SPH. In the experiment the tip is not as

sharp as in SPH, at the left the water level goes up towards the wall because of the drag. In SPH the water level stays horizontal.

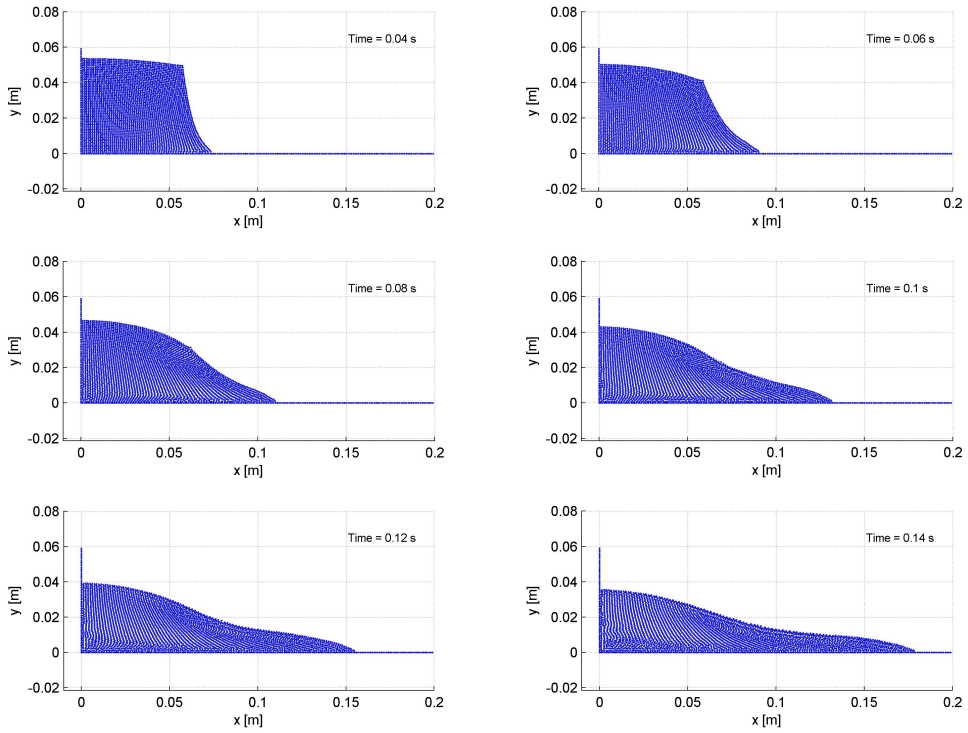


Figure 6.10: Breaking dam.

To compare the results from SPH with the experiment the variables are made non-dimensional with the initial width of the column a. The position of the surge front z becomes $Z = z/a$, the top of the column h becomes $H = h/a$, and time t becomes $T = t\sqrt{g/a}$. During the collapse of a perfect fluid column the following relations hold:

$$Z = F_1[n^2, T], \tag{6.9}$$

$$H = F_2[n^2, T]. \tag{6.10}$$

n^2 denotes the ratio between the original height and width of the column, in this case $n^2 = 1$. In the experiment there were problems finding the right start time of the collapse of the fluid column. In the article **??** the assumption is made, when comparing the measures with a theory, that after $T = 0.1$ motion began. The results of SPH are compared with the originally published experimental data and with the experimental data with adjusted start time.

In figure 6.11 the position of the surge front is plotted as a function of time. The SPH results are very close to the experimental data with shifted start time. The angle of the line indicates the velocity of the tip of the bore. At the start the velocity of the tip is zero, then it increases to a almost constant value between $T = 0.7-2$. Only at the end of the simulation the result of SPH starts to differ significant from the shifted experimental data, probably because of the free slip boundaries in SPH. The contact area between the fluid and the bottom increases during the simulation, so does the bottom friction. In this simulation with SPH the most important friction is internal viscosity, this is not increasing with more contact area between bottom and fluid. This can explain that the velocity of the tip is higher in SPH than in reality at the end of the simulation. But even at $T = 1.8$ the difference between SPH and the shifted experiment is only 3.5 %.

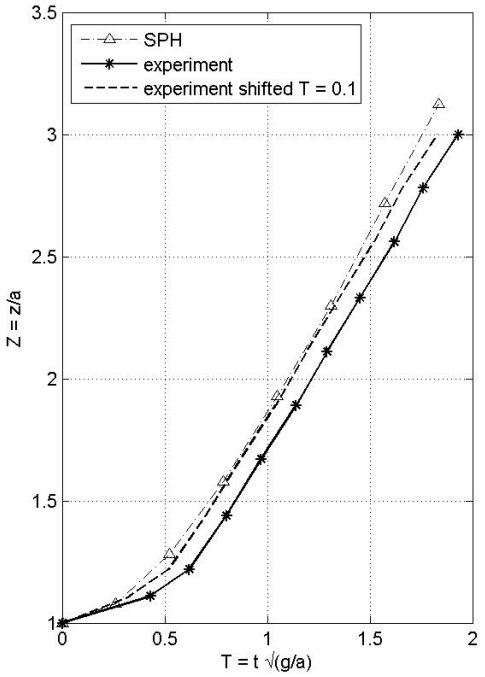


Figure 6.11: Position of the surge front in time.

When looking at the position of the top of the fluid column at the left boundary in figure 6.12 the results of SPH are similar to the results of the experiment shifted $T = 0.1$. Don't look too much at the line of the experiment between the first two stars. Only two measures of h and t are taken, in between just a straight line is drawn. One would expect that the line starts horizontal and then slopes down more, going through the measurement points. From $T = 0.7$ onwards the difference between the experiment shifted and SPH is very small. At $T = 1.8$ the difference is only 0.8 %. Apparently the free slip boundary does have more influence on the horizontal velocity than on the vertical velocity. Given the rather arbitrary chosen time shift of $T = 0.1$, both the position of the surge front and the position of the top of the fluid are simulated correctly with SPH.

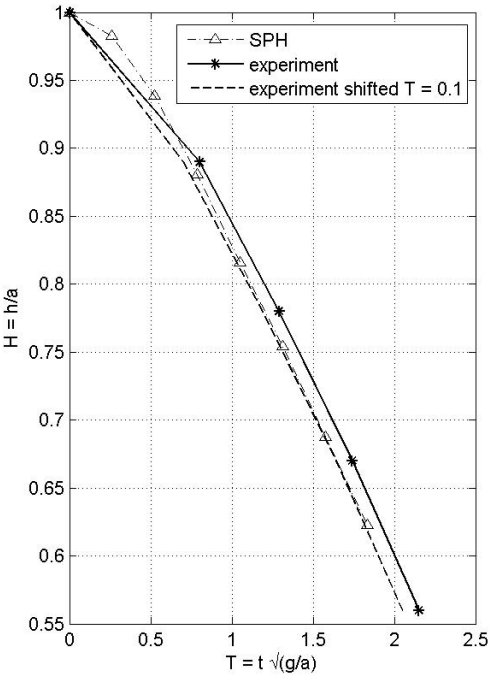


Figure 6.12: Position of the top of the fluid column in time.

6.4 Bore at wall

A horizontal layer of water is suddenly flowing against a vertical wall. The flow is stopped abruptly and a bore will develop, running away from the wall. The long-wave theory cannot be applied to the steep front of the bore, but the theory is valid for the two uniform areas on both sides of the front. See figure 6.13 for the idealised situation considered in the long-wave theory. The conservation of volume per unit of width reads:

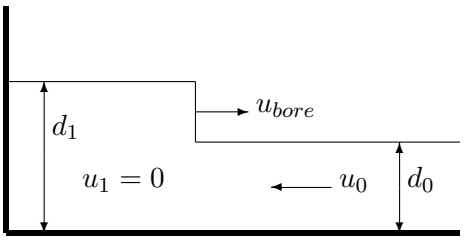


Figure 6.13: Schematic idealised bore.

$$u_{bore}(d_1 - d_0) = u_0 d_0. \tag{6.11}$$

For this idealised stationary situation considering both uniform sides of the bore, the conservation of momentum per unit of width reads:

$$\frac{1}{2}\rho g d_1^2 - \frac{1}{2}\rho g d_0^2 - \rho u_0^2 d_0 = \rho u_{bore} u_0 d_0. \tag{6.12}$$

This equation means that the difference between momentum left and right of the bore on the LHS causes the bore to progress with u_{bore} stopping the movement of water ρu_0 over the whole depth d_0 on the RHS. Combining these two equations and elimination of terms gives:

$$u_{bore} = u_0 + \sqrt{\frac{1}{2} g d_1 (1 + \frac{d_1}{d_0})} \text{ ,with } \frac{d_1}{d_0} = \frac{u_{bore} + u_0}{u_{bore}}. \tag{6.13}$$

This analytical solution can be used to check the water level and velocity of the bore calculated with SPH. This simulation has been carried out with SPH before (Bonet and Lok 1999), the dimensions are chosen according to the earlier simulation. The initial water depth is $d_0 = 0.1$ m. Two simulations have been performed with two different initial velocities $u_0 = 0.2971$ m/s (Froude number = 0.3) and $u_0 = 0.9813$ m/s (Fr = 0.9). The left boundary is fixed, on the right a push boundary is used to give an uniform velocity over the depth flowing into the bore. The total number of fluid particles is constant. The other parameters used, are listed below.

Bore simulation:
349x20 = 6980 particles, $\Delta r = 0.005$ m
Smoothing length $h = 0.007$ m
$c \approx 10\sqrt{2gd} = 14$ m/s
$\Delta t = 0.0001$ s
Artificial viscosity $\alpha = 0.0408$
$\nu \approx \alpha hc/17 = 2.4 \cdot 10^{-4}$ m/s^2
Free slip boundaries
$D = 0.1$ m^2/s

Initial conditions:
$u_x = -u_0$, $u_y = 0$
Hydrostatic pressure

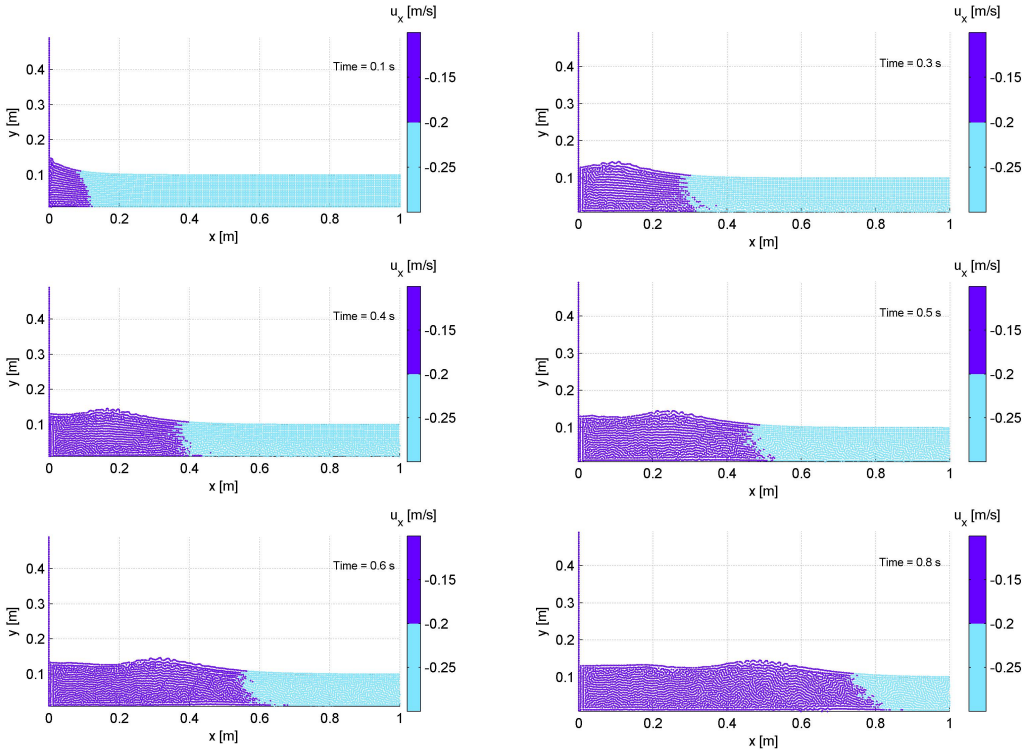


Figure 6.14: Bore 1 with initial velocity $u_0 = 0.2971$ m/s.

In figures 6.14 and 6.15 the particle positions are plotted for six successive times. The color indicates the horizontal velocity. The color is changing at $u_x = -u_0 + 0.1$ m/s, $u_x = -0.2$ m/s and $u_x = -0.9$ m/s for both simulations respectively. This change in color has been used as the front of the bore. Because the different positions of the front in time will be compared with each other, the exact choice of definition for the front of the bore is not that important, as long as the same definition is used every time the front of the bore has to be found. As can be seen in figure 6.14, with $u_0 = 0.2971$ the bore is not very high, and the front of the bore is an unbroken wave. This is called an undular bore. For the second simulation with $u_0 = 0.9813$ m/s the situation is different. The water splashes up the left boundary and gives an over topping wave to the right. After the wave is broken the front of the bore stays turbulent. The tip of the overtopping wave is some particles thick, one would expect that the tip would end in only one particle. This is because the artificial viscosity is too high, but when a smaller viscosity is chosen the simulation becomes unstable. In the last picture of figure 6.15 the right push boundary has just entered the plotted area.

In table 6.1 the average height of the first 0.2 m from the left of the bore is determined. For both simulations these values differs less than 1 % from the theoretic value calculated with the long wave theory. The theoretic velocity of the bore u_{bore} is compared with the value from

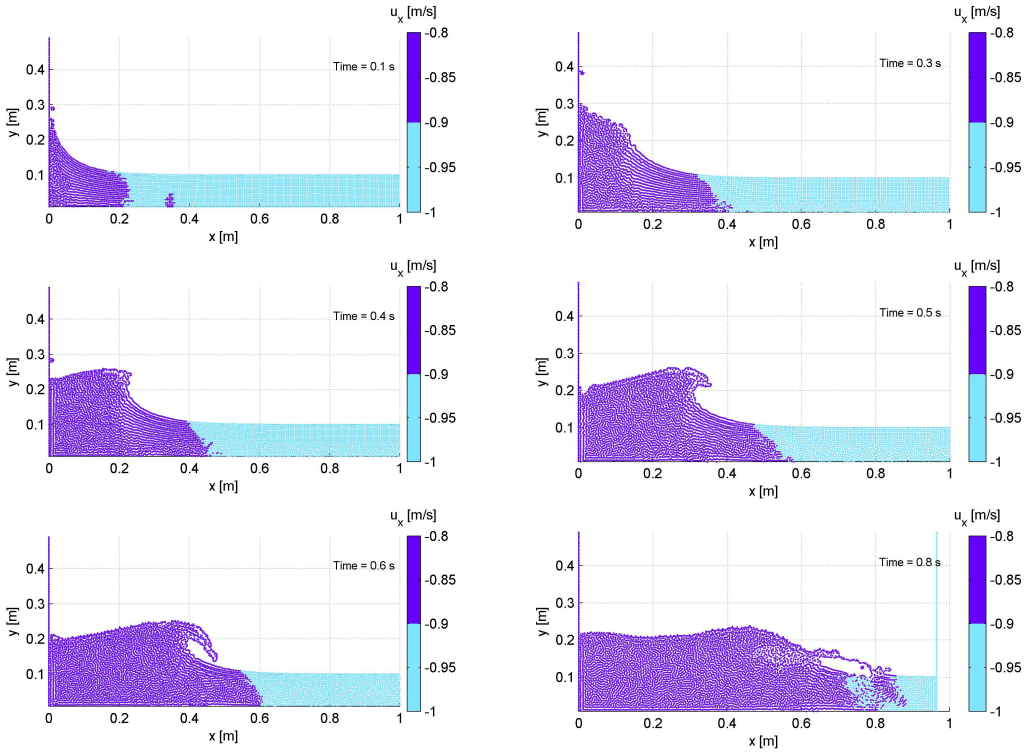


Figure 6.15: Bore 2 with initial velocity $u_0 = 0.9813$ m/s.

x [m]	0	0.05	0.100	0.150	0.200	mean	theory	error
bore 1 d_1 [m]	0.1370	0.1315	0.1317	0.1330	0.1327	0.1332	0.1320	0.9 %
bore 2 d_1 [m]	0.2228	0.2198	0.2165	0.2100	0.2082	0.2155	0.2158	-0.2 %

Table 6.1: Bore height at $t = 0.8$ s for bore 1 ($u_0 = 0.2971$ m/s) and bore 2 ($u_0 = 0.9813$ m/s).

SPH in table 6.2. The velocity of the bore with SPH differs more from the theoretic value than the height of the bore, now the maximum error is 3 %. An explanation could be that the velocity of the bore can not be found accurately from the results. It is found by tracking the front of the bore at two different time steps and dividing the distance the front has moved by the time interval. The front of the bore is found with a visual estimate of the position of the change of color in figures 6.14 and 6.15. Concluding can be said that SPH gives a very accurate prediction of the height of the bore and the velocity seems correct as well, but that is difficult to check exactly.

t	0.3 s	0.8 s	u_{bore}	theory	error
bore 1 front	0.29 m	0.75 m	0.92 m/s	0.929 m/s	-1 %
bore 2 front	0.36 m	0.77 m	0.82 m/s	0.847 m/s	-3 %

Table 6.2: Calculation of u_{bore} for bore 1 ($u_0 = 0.2971$ m/s) and bore 2 ($u_0 = 0.9813$ m/s)

6.5 Standing wave

The way a numerical method is solving the equations can introduce non-physical dissipation of energy. A numerical method can also introduce a phase error in the velocity that information is transfered. The amount of numerical dissipation and the phase error can be tested with the simulation of a periodic process without physical damping. A standing wave in a closed basin with free slip boundaries is such simulation, without friction the total amount of energy, this is the sum of potential and kinetic energy, will stay constant and the wave period is determined by the geometry. With numerical dissipation the total amount of energy will decrease, with a numerical phase error the wave speed in the model will differ from the physical wave speed leading to a different wave period. A closed basin of effective 1.975 m wide with a cosine-shape standing wave of 4.95 m long is used. Exactly half a wave length fits in the basin, the initial amplitude H is 0.15 m. The water depth d is 2 m. The wave is characterised by:

$$\begin{gathered} k = 2\pi/L = 2\pi/3.95 \text{ m} = 1.59 \text{ rad/m}, \\ \omega^2 = gk \tanh(kd) = 15.55 \text{ rad}^2/\text{s}^2 \rightarrow \omega = 3.92 \text{ rad/s}, \\ T = 2\pi/\omega = 1.59 \text{ s}, \\ H/d = 0.075, \ H/L = 0.03. \end{gathered}$$

The linear wave theory predicts that the shape of the wave is not changing over time, but the linear theory is only valid in cases where the amplitude is much smaller the the water depth. In this simulation the ratio amplitude/depth is 0.075, maybe some non-linear effects are visible at the end of the simulation. Next to the visible changes in the water level this standing wave problem can have other evanescing solution modes which distract energy from the visible surface elevation. From the surface elevation itself no hard conclusions can be drawn, but also with small non-linear influences and evanescing solution modes the total amount of energy in the system should stay constant.

6.5.1 Standing wave with standard boundary force

In order to keep the simulation stable SPH needs some artificial viscosity. In this simulation it appeared that minimal $\alpha = 0.01$ leading to $\nu \approx \alpha hc/17 = 1 \cdot 10^{-3}$ m/s^2 was necessary. This artificial viscosity dissipates energy. Another source of dissipation is the radial boundary force around every boundary particle to give a closed boundary. The flow parallel to boundaries does feel this radial boundary force as ripples, the ripples dissipate energy as well. Because of the combination of a boundary force with ghost particles the boundary force can stay small to ensure a closed boundary, but the dissipation from the boundary force is not zero. The wave speed can be calculated by $u_{wave} = \omega/k$, in this simulation the wave number k is fixed by the geometry. Dissipation will cause the wave to slow down leading to a lower u_{wave}, with constant k this gives a lower ω. A lower ω is equal to a higher period T. The conclusion is that dissipation will increase the wave period. SPH introduces some dissipation from the artificial viscosity and the radial boundary force, therefore it is expected that the wave period

in SPH is larger than 1.59 s.

The initial conditions in this simulation are chosen according to the wave in its middle, with zero amplitude and maximum velocities. See figure 6.16 for a visualisation. The initial velocities are calculated with the linear wave theory. Starting with zero amplitude gave better results than starting at a maximum amplitude because the position of the particles is discrete and the initial velocity given to the particles can be any value. The most important numerical parameters are:

Standing wave simulation:
79x80 = 6320 particles, $\Delta p = 0.025$ m
Smoothing length $h = 0.035$ m
$c \approx 11\sqrt{gd} = 50$ m/s
$\Delta t = 0.0002$ s; $\Delta t/T = 1/7950$
Artificial viscosity $\alpha = 0.01$
$\nu \approx \alpha hc/17 = 1 \cdot 10^{-3}$ m/s^2
Free slip boundaries
$D \approx 1/2gH = 10$ m^2/s

Initial conditions:
Horizontal water line
Velocities according to standing wave
Hydrostatic pressure

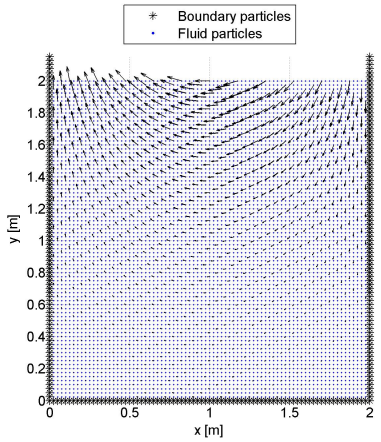


Figure 6.16: Initial particle positions standing wave with initial velocity quiver.

The top graph in figure 6.17 shows the elevation of the surface at $x = 0.1$ m, the line is the linear theoretic elevation for a wave without dissipation, the stars are the results from SPH. The amplitude in SPH is decreasing over time, due to the artificial viscosity and the boundary force, but non-linear effects are absent in these 4 periods. The numerical dissipation in SPH leads to a lower wave speed and a higher period, but this higher period is not visible in the results. Next to numerical dissipation as a source for wave speed errors also a numerical method itself can give wave speed errors. It appears that the numerical translation in SPH of the original equations introduces a wave speed which is too big. The positive phase error from the numerical method itself cancels out against the negative phase error from the numerical dissipation. The wave in SPH does have the same period as the theoretic wave, and the amplitude is decreasing.

The amount of dissipation of energy can be seen in the development the energy in the lower graph of figure 6.17. From the potential energy the initial potential energy is subtracted, the total energy is the sum of the potential and kinetic energy. The fluid settles about 6 mm at the start which destroys potential energy. This settlement causes the potential energy to become under the initial potential energy, this gives negative values in the graph. The kinetic energy is maximal when the potential energy is minimal, and the line of the total energy is

almost straight. So the conversion between kinetic and potential energy is correct. Because of the settlement of the fluid not the total energy is used to find the amount of dissipation, but the maximum kinetic energy in a period.

Without dissipation the maximum kinetic energy in a period should stay constant during the simulation. The initial kinetic energy in SPH is 104 J/m, this is a little less than 109 J/m expected from theory. The reason for this is the discrete positions and velocities of the particles in stead of the theoretic continuous fluid and velocity field. After four periods only 32 J/m of kinetic energy is left, this is 31 %. The artificial viscosity and closed boundaries in SPH are dissipating a lot of energy, 69 % in four wave periods.

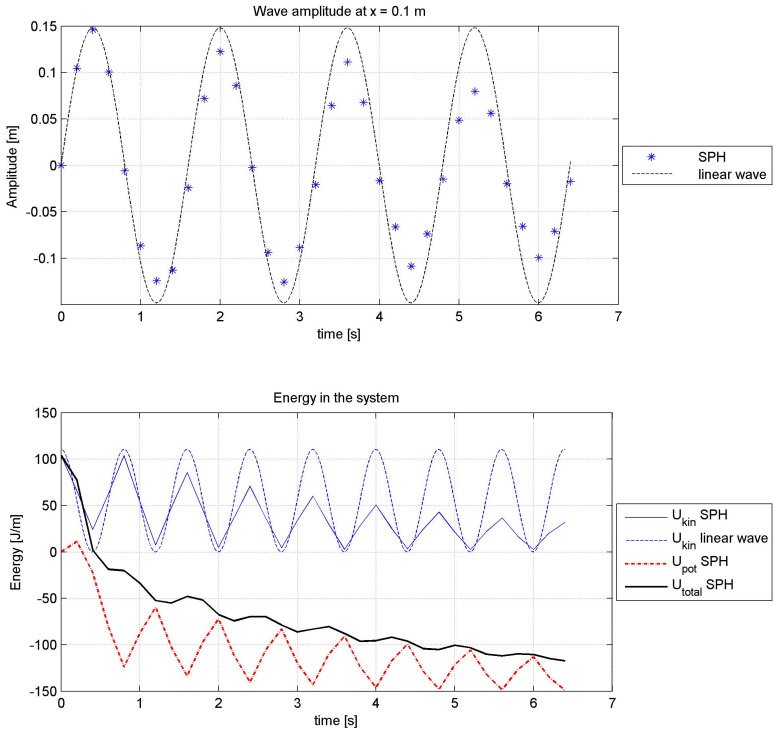
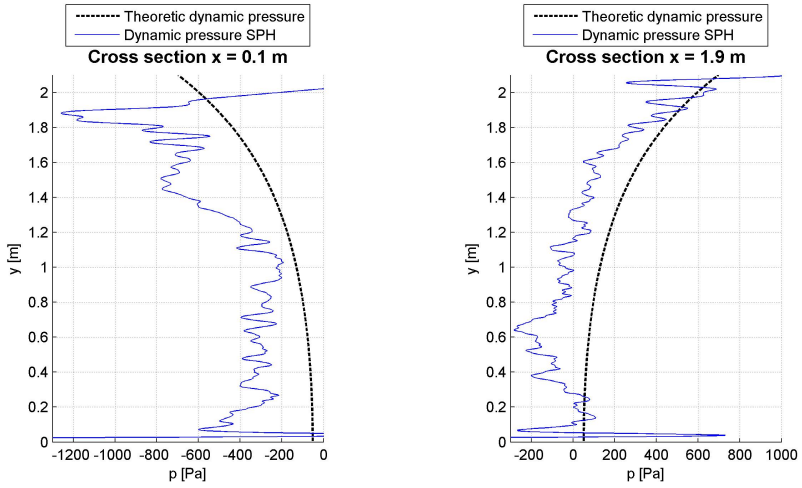


Figure 6.17: Time series of standing wave in SPH with radial boundary force.

The dynamic pressure in SPH at $t = 6$ s at both ends of the standing wave in its extreme elevation is shown in figure 6.18. The dynamic pressure is the difference between the linear hydrostatic pressure and the actual pressure in a wave. The theoretic dynamic pressure is calculated using the linear wave theory with the decreased amplitude at $t = 6$ s of 0.09 m. The shape of the dynamic pressure in SPH is correct, it is negative when the surface elevation is negative and positive when the elevation is positive. The dynamic pressure in figure 6.18(b) is close to the theoretic expected pressure, but in figure 6.18(a) the difference at some points is more than 100 %. Maybe this comes from the influence of non linear wave components which start to evolve, or some compression waves walk through the basin.



(a) Pressure negative elevation.

(b) Pressure positive elevation.

Figure 6.18: Dynamic pressure in standing wave at $t = 6$ s.

6.5.2 Standing wave with adapted boundary force

The artificial viscosity is necessary to keep the simulation stable, but the closed boundary could be adapted a little to remove the ripples of the boundary force. This would reduce the numerical dissipation in SPH. The radial boundary force per unit of mass around every boundary particle is defined with following equation from section 4.2.1:

$$f(r_{ij}) = \begin{cases} D\left((\frac{r_0}{r_{ij}})^{12} - (\frac{r_0}{r_{ij}})^4\right)\frac{\mathbf{x}_{ij}}{r_{ij}^2} & r_{ij} \leq r_0 \\ 0 & r_{ij} > r_0 \end{cases} \tag{4.10}$$

with r_{ij} defined as the distance between a fluid particle and a boundary particle and length scale r_0 taken equal to the initial particle spacing Δr. The complete explanation can be found in section 4.2.1. This boundary force gives a bumpy boundary force field and the flow along the boundary does feel this as ripples. To remove the ripples, the boundary force is changed from radial around every boundary particle to a direction always square at the boundary. This is achieved by changing the boundary force into:

$$f_y(\Delta y) = \begin{cases} \frac{D}{\Delta y}\left((\frac{r_0}{\Delta y})^{12} - (\frac{r_0}{\Delta y})^4\right) & \Delta y \leq r_0 \\ 0 & \Delta y > r_0 \end{cases} \tag{6.14}$$

f_y is the vertical boundary force at the bottom. Δy is the distance between a fluid particle and the bottom, r_0 and D are the same as in the original boundary force. Near the bottom only a vertical boundary force is active. Near a vertical wall simply change every direction y in this boundary force into x and the horizontal boundary force is found. This boundary force doesn't give ripples and is not dissipating energy of the flow flowing parallel to a boundary.

The results of the standing wave in SPH with this new boundary force is displayed in figure

6.19. The surface elevation is displayed in the top graph. Because there is less dissipation, the wave speed will be higher than in the last simulation, the wave period will be smaller. The top graph shows that the sinus-shaped elevation in SPH at the end of the simulation is shifted to the left compared with the linear wave. The period in SPH is indeed smaller than 1.59 s last simulation. The graph of the energy in the system will be used to find the amount of dissipation. The settlement of the fluid at the start can again be seen in the line of the potential energy. At the end of the simulation, the line of the total energy is not straight, but there are humps. The maximums of the kinetic energy do not correspond to the minimums of the potential energy. Next to the kinetic and potential energy, there must be another type of energy which is important here. Maybe a compression wave walks through the basin and exchanges energy with the potential energy as well.

The phase shift in the surface elevation is also visible in the line of kinetic energy, at the end the data points from SPH do not correspond with the maximums anymore. To find the maximum kinetic energy at the end a smooth spline function is fitted on the results. At the end 44 J/m is left of the original 104 J/m, this is 42 %. The dissipation in SPH is 58 % over 4 periods, much less than 71 % dissipation in last simulation with the radial boundary force. The adapted boundary force does removes the friction at the wall and reduces the total dissipation of energy a lot. But it is not easily applicable at irregular boundary shapes. This boundary force is only used for this simulation, maybe it could improve results of some other simulations as well.

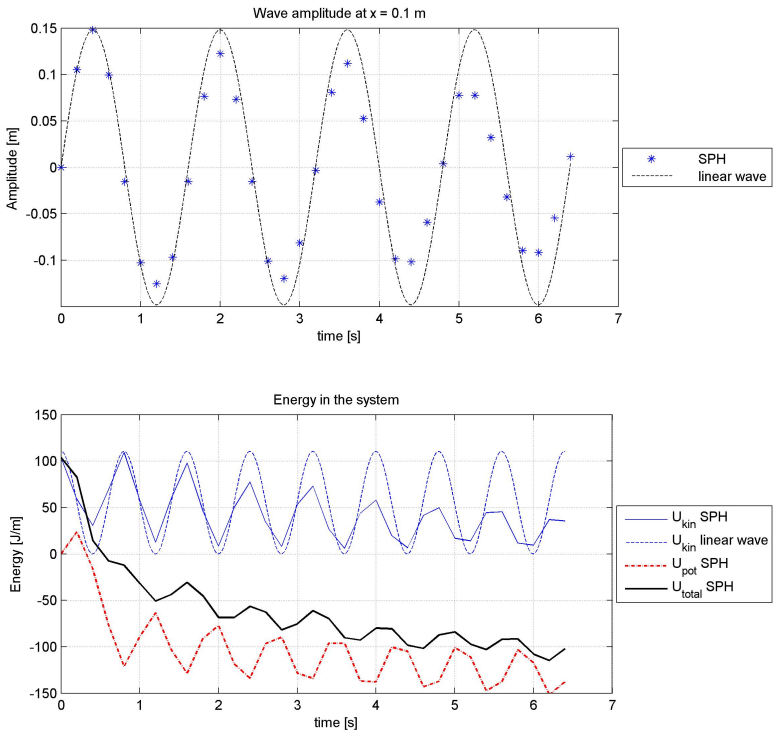


Figure 6.19: Time series of standing wave in SPH with perpendicular boundary force.

6.6 Wave propagation on a beach

This simulation is about the propagation of waves on a beach. On the left a vertical wave maker generates waves, there is a beach with a slope of 1:10 on the right. Between the utmost left position of the wave maker and the start of the beach is a horizontal section of 10 m. With the wave maker regular waves are generated with constant amplitude and period. Two simulations with two different wave periods on the same beach are presented. They will show clearly distinct breaking behaviour. The way a wave is breaking on a beach depends on the steepness of the wave and the slope of the beach, this is combined in the parameter of Iribarren, explained in the lecture notes of Battjes (2001), which is given by:

$$\xi = \frac{\tan \alpha}{\sqrt{H/L_0}}, \tag{6.15}$$

where α is the slope of the beach, H is the waveheight at the breaking point and L_0 is the wavelength at deep water. When $\xi < 0.4$ waves will spill giving some foam at the crest, but no overtopping. When $0.4 < \xi < 2$ waves will plunge. Plunging means the wave top goes faster than the wave itself, resulting in an overtopping crest. The crest hits the base with a lot of energy giving a high splash. In the first simulation the wave generator creates waves with $\omega = 2$ rad/s, $T = 2\pi/\omega = 3.14$ s. $L_0 = gT^2/(2\pi) = 15.4$ m. The stroke of the wavemaker is 1.5 m, this results in a maximum waveheight of $H = 1.84$ m. Iribarren's parameter is $\xi = 0.29$, the waves will spill. The second simulation is performed with much longer waves on the same beach. $\omega = 1$ rad/s, $T = 2\pi/\omega = 6.28$ s. $L_0 = gT^2/(2\pi) = 61.6$ m. The stroke of the wavemaker is now 2.5 m, resulting in a waveheight $H = 2.4$ m at the breaking point. $\xi = 0.51$, now the waves will plunge and show overtopping.

In SPH the wave maker consists of a vertical row of boundary particles with a prescribed horizontal velocity and position according to a sinus-shape movement. With ghost particles created at the wave maker sometimes fluid particles are glued high at the waveboard and fall down later on. This disturbs the fluid motion more than a small error in the pressure near the wave maker, therefore ghost particles are not created at the left boundary. At the other boundaries ghost particles are created as usual. Fluid particles of 0.1 x 0.1 m are used, for overtopping waves with a waveheight $H \approx 2$ m this gives enough resolution to capture the overtopping. Initially the water level is horizontal with a depth of 5 m in the horizontal section. About 6 wave periods are simulated, resulting in 20 s for the first and 40 s for the second simulation. Calculation time was 16 and 31 hour respectively on a 2 GHz PC with 1 gb memory under Linux, the calculation time was the main reason not to model more wave periods. The results are sensitive to the amount of artificial viscosity. With too little viscosity particles go everywhere, with too much viscosity the waves are smeared out and cannot have a sharp crest, overtopping is certainly impossible then. The viscosity used in these simulations was found after many trials and is the lowest viscosity possible without large disturbance of the free water surface. The parameters used in SPH for these simulations are summed up now.

Wavemaker simulation:
17192 particles, $\Delta p = 0.1$ m
Smoothing length $h = 0.14$ m
$c \approx 7\sqrt{gd} = 50$ m/s
$\Delta t = 0.001$ s
Artificial viscosity $\alpha = 0.02$
$\nu \approx \alpha hc/17 = 8 \cdot 10^{-3}$ m^2/s
Free slip boundaries
$D = 10$ m^2/s

Initial conditions:
$u_x = 0$ m/s, $u_y = 0$
Hydrostatic pressure

Left boundary:
Wave maker
Right boundary:
Beach with 1:10 slope

6.6.1 Spilling waves

The first simulation is with short waves which will spill on this beach because $\xi = 0.29$. The particle configuration after 19 s is showed in figure 6.20. The crests are sharp and when a movie of the progressing waves is analysed it is visible that the crests crumble and do not overtop. The shape is as expected with spilling waves. The length of the first complete wave visible at the left is around 15.7 m, this is more than the deep water wavelength $L_0 = 15.4$ m. In reality the wave slows down towards the shore because of the decreasing depth, and the wave period stays constant. From the relation $u_{wave} = \omega/k$ the conclusion is that with constant ω a wave gets shorter when the wave speed decreases. From the dispersion relation the wave length at every depth can be calculated when the wave period and the depth are known, for the first wave the length should be 14.3 m. Next wave is 13.6 m long when 12.4 m was expected from theory. Last wave is 11 m long when 10 m was expected. Also in SPH the waves slow down and the wave length decreases with decreasing depth, the influence of the depth on a wave is correct in SPH. But the absolute value of a wave length in SPH is approximately 10 % too long. A possible explanation comes from the positive phase error in the numerical integration of SPH as already found in the standing wave simulation. A positive phase error leads to a wave speed which is too high and this gives a wave which is too long.

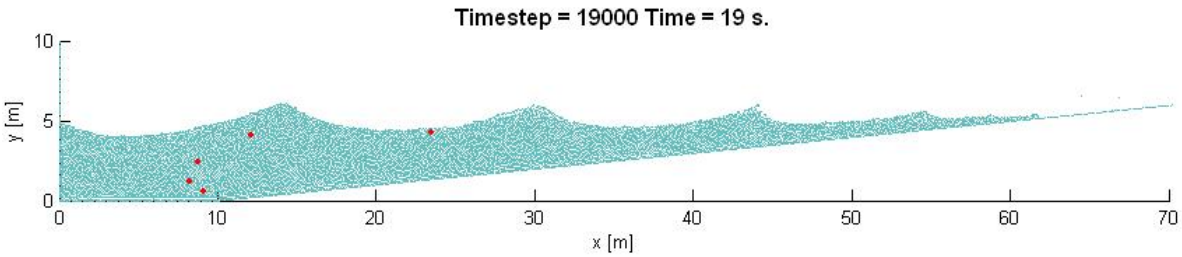


Figure 6.20: Spilling waves on the beach.

The path of five particles, indicated with dark dots in figure 6.20, is followed during the simulation. At $t = 0$ they all are at $x = 10$ at different vertical positions. In figure 6.21 their paths

are shown. Under influence of the progressing waves the particles make an orbital motion, which is an ellipse with the longest axis in the horizontal direction. They do not move in a closed ellipse, but every period they have a net displacement. This is called the Stokes' drift and is also seen in experiments. The vertical displacement in the ellipse is larger for particles who are further away from the bottom. Near the bottom the ellipse is almost flat, near the free surface the ellipse is more round. The lowest particles follow the slope of the beach. The net displacement is to the right for the highest two particles and to the left for the lowest three particles. At the top of a progressive wave the mass transport is in the direction of propagation and at the bottom there is a counter current. In the final state the nett mass transport over the vertical is zero. The orbital motion of the particles in SPH is as expected in a progressing wave.

The conclusion about this spilling wave simulation on a beach is that SPH can handle this type of waves. Physical phenomena like crumbling spilling wave crests and an elliptical orbital motion are visible. The sharp crest and non hydrostatic pressure in a short wave was no problem. Unfortunately the predicted wave length was too large, but with another integration scheme or smoothing kernel, with a smaller phase error, this probably can be improved.

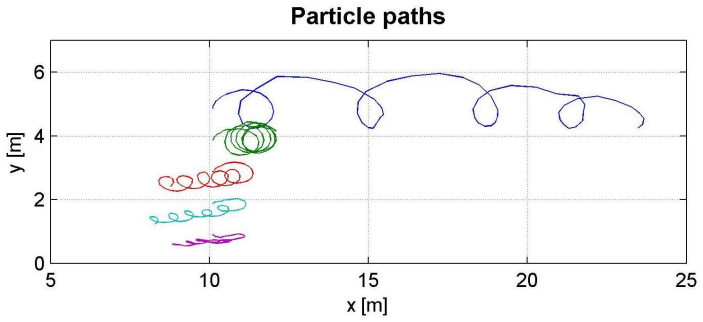


Figure 6.21: The orbital motion of 5 particles under spilling waves.

6.6.2 Plunging waves

Second type of waves generated with the wave maker have a period twice as large as previous waves. Their length in deep water is much bigger $L_0 = 61.6$ m, their height is comparable. These longer waves feel the same beach as much steeper than the previous shorter waves. From $\xi = 0.51$ it can be concluded that these waves will plunge on the beach, showing an overtopping and breaking character as known from the Dutch beach. The particle configuration after 13.5 s in figure 6.20 shows an overtopping wave. Starting at the wavemaker every wave moves to the right. When the waves enter shallower water shoaling becomes visible. Shoaling means that a decreasing depth leads to a lower wave speed, a shorter wave length and an increasing

wave height. This process continues until the wave becomes unstable and the wave breaks. A close up with six snapshots of a breaking wave is shown in figure 6.23, the color indicates the original horizontal position. The crest of the wave curls over more and more until it falls into the base. This gives a big splash. The different colors in figure 6.23 show there is some mixing after breaking, but it is smaller than in real waves. The artificial viscosity needed for stability is too high to show more realistic behaviour with more vorticity, and finer splash up. But SPH can capture the overtopping itself and that is spectacular already.

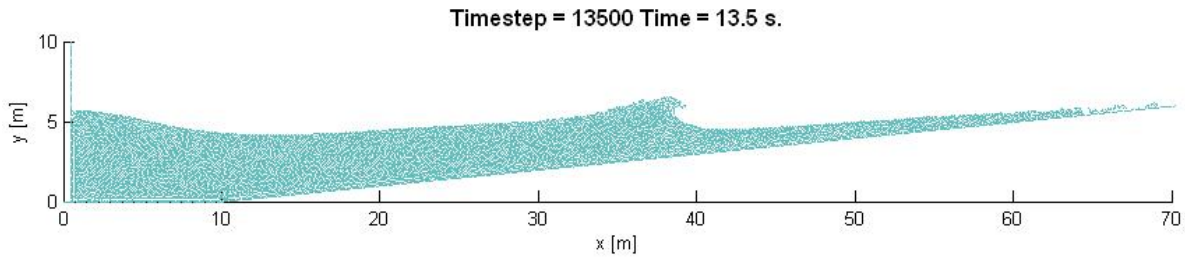


Figure 6.22: Plunging wave on the beach.

From time-series of the water level at certain points 5 m from each other the time averaged level of the wave crest, wave trough, and mean water level is determined and displayed in figure 6.25. The time averaged values are only from 2-4 wave periods depending on the distance from the wave maker. The wave crests become higher until the break point at $x = 35$ m, at $x = 40$ m the wave is overtopping. The displayed averaged wave crest level at $x = 40$ m is the lowest free surface level under the tube of the overtopping wave. At $x = 45$ m the crest has fallen and the crest hight is decreased a lot. After breaking the crest level stays constant towards the shore. The wave trough increases a little towards the shore until the break point, after the break point it increases faster. From theory and measurements it is known that the mean water line goes down a little when going from deep water to shallow water until the break point. This is called the wave set-down, a rough theoretic value is given by (Battjes 2001):

$$\bar{\eta} = -1/8 \frac{kH^2}{\sinh\, 2kd}, \tag{6.16}$$

where k is the wave number, H is the local wave height inclusive shoaling effects, and d is the local waterdepth. The theoretic wave set-down is calculated with reference to the still water line, indicated with s.w.l. in figure 6.25. At sea the s.w.l. is the mean water level at deep water, in this small scale simulation it can be calculated from conservation of volume in the basin. The still water line in this simulation is at $y = 5$ m. Before the break point the mean water level in SPH is under the still water level, some set-down is visible. Towards the break point the set-down in SPH becomes less than the theoretic value, but from measurements is known that this theoretic value of the set-down is too large at the break point. The exact set-down can only be compared with an experiment.

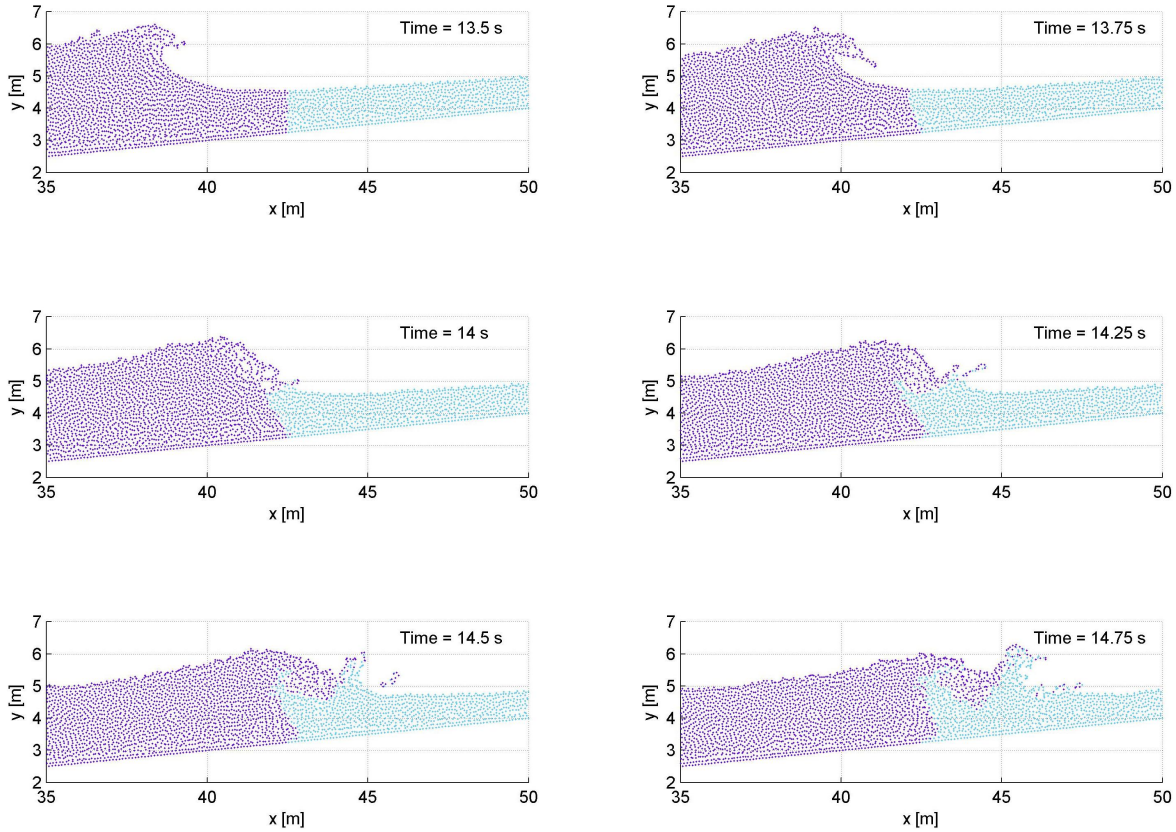


Figure 6.23: Close up of a breaking wave, the color indicates the position in the first snapshot.

More important than the set-down before breaking is the set-up after breaking. This wave set-up after the break point is bigger than the set-down and it determines how far waves can propagate on the beach. The run up of waves determines the minimal hight a sea defence has to have to prevent overtopping. In SPH there is a big set-up after the breaking point, much more than the set-down before breaking. This is just as expected. A rough estimate of the slope of the mean water level after the breaking point is given by (Battjes 2001):

$$\frac{d\bar{\eta}}{dx} = \frac{3/8\gamma^2}{1 + 3/8\gamma^2}\tan\,\alpha, \tag{6.17}$$

where γ is the breaking index defined by H_b/depth and H_b is the wave height at the breaking point. α is the slope of the beach. The breaking index is dis-

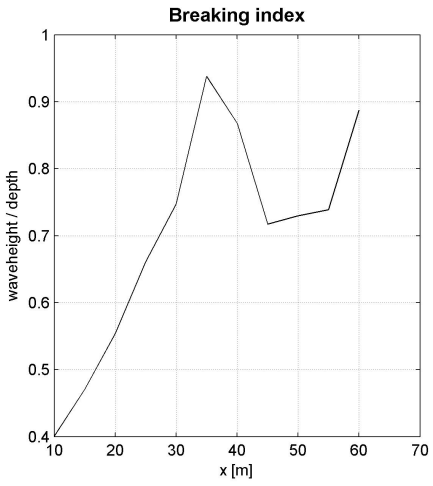


Figure 6.24: Breaking index γ for plunging waves along the shore.

played in figure 6.24. In this simulation $\gamma = 0.94$ at breaking, the theoretic set-up line is drawn in figure 6.25. The mean water level in SPH for $x > 35$ m is close to the theoretic set-up line. When the breaking index along the shore is analysed from figure 6.24, it is clear that the waves grow until the break point at $x = 35$ m with $\gamma = 0.94$. From measurements it is known that plunging waves break between $\gamma = 0.8$ and $\gamma = 1.2$, the value from SPH is within this range. After the break point γ decreases to 0.73 and increases again to the breaking value at $x = 60$ m.

The conclusion about this simulation is that SPH can also handle plunging waves on a beach. Phenomena like shoaling, a small wave set-down before the break point and a big wave set-up after breaking are present in the results of SPH. The breaking of waves itself is captured from the Navier-Stokes equations without further assumptions. The results from SPH are realistic, but the exact values can only be verified with an experiment.

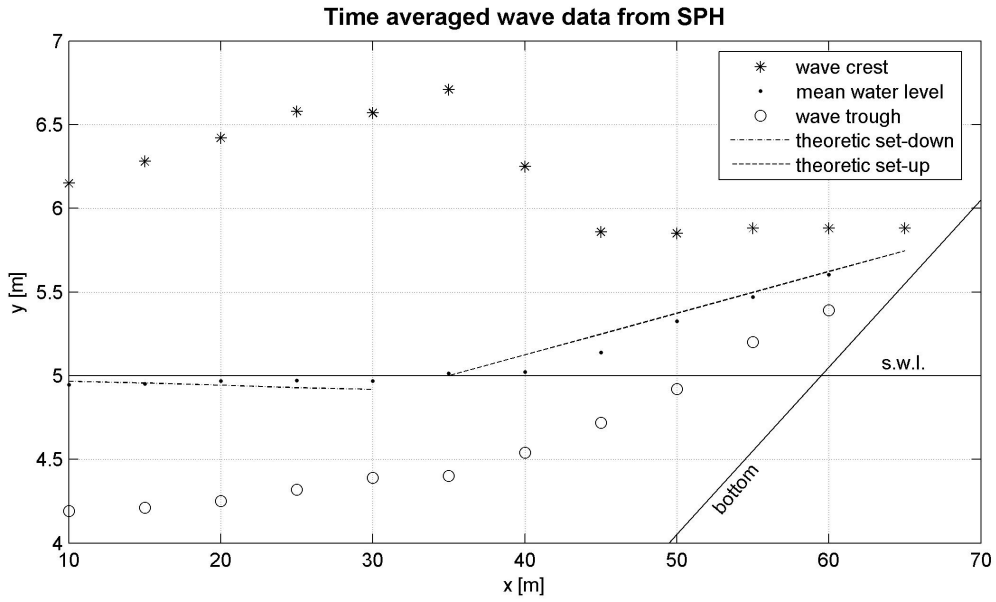


Figure 6.25: Comparison of wave data from SPH with linear theory.

6.7 Sharp weir

When water is flowing over a sharp weir, like drawn in figure 6.26, only one measure of the waterlevel is enough to know the discharge over the weir. Therefore this type of weir is used a lot to measure discharges in laboratory flumes or irrigation channels etc. At the weir the waterlevel is going down fast and the pressure gradient is large. This is the reason that many other numerical methods cannot handle a sharp weir. The aim of this section is to show that SPH can handle a sharp weir.

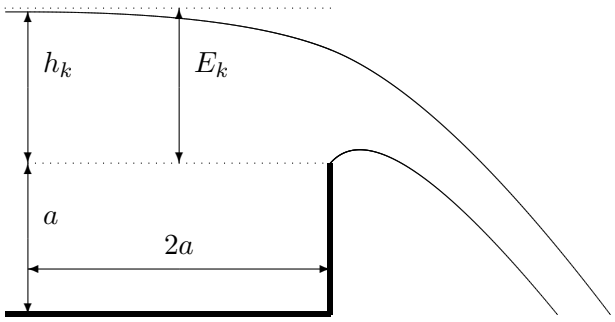


Figure 6.26: Schematic idealised sharp weir.

At a clear overfall the water depth at the overfall is 2/3 of the energy level E_k and the depth averaged velocity at a clear overfall is equal to $\sqrt{2/3gE_k}$. Rehbock combined this theory with some empirical adjustments. The energy level E_k is almost equal to the much easier to measure water level h_k, first Rehbock replaced E_k with h_k and then he added an empirical correction factor to h_k, leading to $h_e = h_k + 0.11$ cm. The empirical correction of 0.11 cm is needed to take viscosity and surface tension into account. He used h_e to calculate the discharge over a sharp weir:

$$q = m' \frac{2}{3} h_e \sqrt{\frac{2}{3} g h_e}, \tag{6.18}$$

where q is the discharge per unit of width, $g = 9.81$ m/s^2. m' is a discharge coefficient given by:

$$m' = 1.045 + 0.141 \frac{h_e}{a}. \tag{6.19}$$

The water level with regard to the weir height h_k has to be measured $2a$ upstream, with a defined as the weir height. The discharge can be calculated with only one measure of a waterlevel, all other parameters are known for a given weir. The factor m' is larger than 1 because of the curvature of the streamlines above the weir.

In SPH the sharp weir consists of a vertical line of fixed boundary particles. All particles that fall below $y = 0$ right of the weir are removed, at the left an inflow boundary with constant waterlevel is modelled. The details about this inflow boundary are mentioned in section 4.2.3, the most important used parameters in this simulation are mentioned now.

Sharp weir simulation:
1300 particles on average, $\Delta p = 0.025$ m
Smoothing length $h = 0.035$ m
$c = \approx 11\sqrt{gd} = 22$ m/s
$\Delta t = 0.0005$ s
Artificial viscosity $\alpha = 0.01$
$\nu \approx \alpha hc/17 = 5 \cdot 10^{-4}$ m/s^2
Free slip boundaries
$D = 0.1$ m^2/s

Initial conditions:
$u_x = 0.5$ m/s, $u_y = 0$
Hydrostatic pressure

Left inflow boundary:
waterlevel = 0.4 m
Right sharp weir:
$a = 0.175$ m

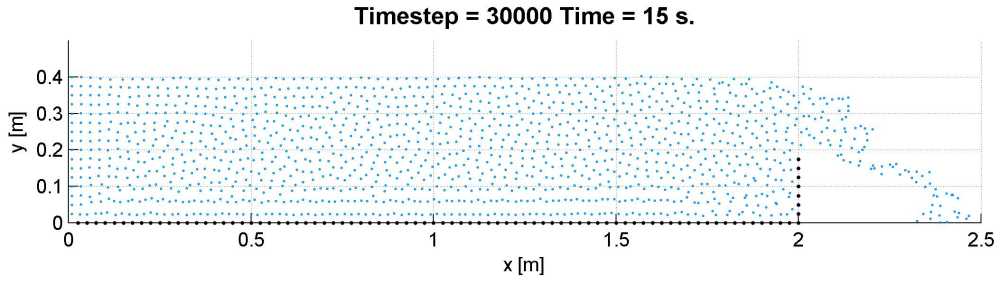


Figure 6.27: Flow over a sharp weir.

The flow over a sharp weir in SPH from figure 6.27 gives the same impression as the schematic idealised flow over a sharp weir in figure 6.26. Note the regular position of the particles at the inflow. Near the bottom some lines with fluid particles with a small horizontal distance and a large vertical distance are visible. This is some negative influence of the bottom boundary. The waterlevel after the inflow is almost horizontal, over the weir the flow bends down and gets a free fall. The XSPH technique from section 4.3 is used to move the particles. Only for particles that are over the weir the vertical velocity to move the particle is not adjusted with XSPH, because then the falling flow after the weir would pull the flow left from the weir and this gives bad results. Outside the weir ghost particles are generated to give the correct pressure near the weir. Ghost particles are generated for every fluid particle left of the weir and lower than the influence depth of $y = 0.2$ m. These ghost particles can only interact with fluid particles left of the weir, the fluid particles who are already over the weir cannot interact with the ghost particles.

In order to compare the results of SPH with the Rehbock formula (equation 6.18) a reference level at the bottom of the flow is introduced and the exact height of the weir is determined. As explained in section 6.1 every particle represents an area of $(\Delta r)^2$. The bottom of the flow is not at $y = 0$ but $1/2\Delta r$ below the centre of the lowest particles. From the results is found that the bottom of the flow is at $y = 0.0125$, this is defined as the reference level. The sharp weir consists of a row of vertical fixed boundary particles until $y = 0.175$ m. The boundary force of the top boundary particle of the sharp weir has an influence distance of $r_0 = \Delta r = 0.025$ m. The lowest particle that can flow over the weir is at $y = 0.175 + 0.025 = 0.2$ m, the lowest

particle also represents fluid $1/2\Delta r$ under its centre, therefore the effective top of this sharp weir in SPH is at $y = 0.2 - 1/2\Delta r = 0.1875$ m. According to the reference level the sharp weir is effectively $0.1875 - 0.0125 = 0.175$ m high.

In summary, from every vertical coordinate y the reference level of $y = 0.0125$ m has to be subtracted to get the effective depth. At the left there is an inflow boundary with a constant fixed water level of $y = 0.4125$ m, this corresponds to an effective depth of $0.4125 - 0.0125 = 0.4000$ m. This is correct because the inflow consists of a vertical row of 16 particles each representing 0.025 m, and $16 \times 0.025 = 0.4000$ m. At the right there is a sharp weir of effectively 0.175 m high.

x [m]	0	0.5	1.0	1.5	2.0
depth [m]	0.3998	0.3940	0.3926	0.3918	0.3661

Table 6.3: Time averaged water depth sharp weir (reference $y = 0.0125$ m).

h_k	h_e	m'	$q_{Rehbock}$	q_{SPH}	error
0.2186 m	0.2196 m	1.222	0.2145 m^2/s	0.2260 m^2/s	5.4 %

Table 6.4: Discharge sharp weir from Rehbock's formula.

All quantitative results showed here are averaged over the last 5 s of the simulation. The waterlevel, pressure and discharge at a position are varying in time. The simulation never gets into a perfect equilibrium, but it varies around it. First the water level will be analysed. There is some negative influence of the inflow boundary left, the waterlevel drops around half a centimeter directly, this is visible in figure 6.27. In table 6.3 the averaged depths show that between $x = 0.5$ m and $x = 1.5$ m the waterlevel only drops 2 mm. Because of the viscosity some energy is dissipated, leading to a decreasing energy level towards the right. When the energy level decreases also the water depth has to decrease. The Rehbock discharge is calculated in table 6.4, using the water level h_k at $2a = 0.35$ m upstream of the weir. The discharge in SPH is determined by counting the volume of the inflow particles at the left. The discharge in SPH is 5.4 % too high.

The time averaged pressure near the weir is showed in figure 6.28. The hydrostatic pressure at the inflow on the left is shown with the thick black line. At the weir the pressure is expected to be totally different. At the free surface ($y = 0.3786$ m) the pressure should be the atmospheric pressure (p=0). Also at the top of the weir ($y = 0.2$ m) the pressure should be atmospheric because the flow is ventilated. The pressure at the bottom of the weir is expected to have the value of the hydrostatic pressure at the bottom near inflow, because it is a stagnation point. The expected pressure at the weir goes through the points just mentioned, but its shape between the points is just qualitatively.

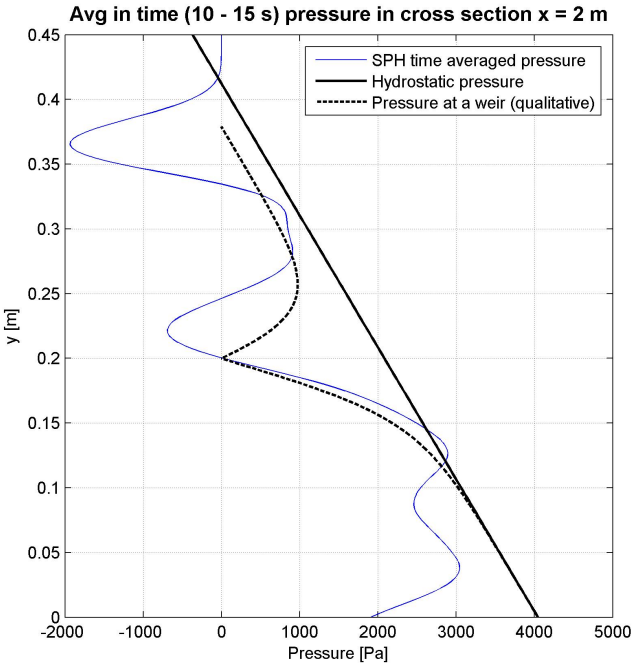


Figure 6.28: Pressure at the weir.

When this expected pressure at the weir is compared with the time averaged pressure in SPH, the agreement is not bad at all. At the top of the weir the pressure in SPH is zero, in SPH also the flow is ventilated. Towards the bottom the pressure increases to hydrostatic pressure. Near the stagnation point at the bottom the pressure in SPH is too small, because it is in a corner with two rows of boundary particles which do not contribute to the pressure. In the top of the flow the pressure in SPH differs from the expected pressure, it is not zero but negative.

SPH can handle the large pressure differences and bending water line at a sharp weir, the results are close to the theoretic expected results. The discharge differs only 5.4 % and the time averaged pressure at the weir looks realistic.

Chapter 7

Conclusions and recommendations

7.1 Overview

The Smoothed Particle Hydrodynamics method has been studied and a 2D version of this method is programmed in Fortran. The influence of different implementations in the code is analysed. To find out what the prospects of SPH are in hydraulic engineering several 2D vertical simulations with a free surface have been carried out. These simulations comprise the water motion at a breaking dam, a bore at a wall, a standing wave, waves propagating on a beach and a sharp weir. Also some 2D horizontal problems have been carried out to test the artificial viscosity, these are Poisseuille flow, Couette flow and a shear driven cavity problem. In this chapter the conclusions about the application of SPH in hydraulic engineering are presented together with some recommendation to improve SPH.

7.2 Conclusions

- SPH in general
  SPH is a Lagrangian particle method which can solve the total Navier-Stokes equations. Because SPH uses moving particles, it does not have troubles with advection. The fluid is made slightly compressible ($\Delta\rho/\rho \leq 1\%$), to get a time derivative of the density. The pressure is calculated with an explicit equation of state from the density, and can be non-hydrostatic.

- Viscosity
  The influence of viscosity is not modelled with exact second derivatives of the velocity, but approximated with an artificial viscosity term. In the shear driven cavity problem, the artificial viscosity underestimated the maximums in the velocity. Poisseuille and Couette flow did result in the correct velocity profile. The artificial viscosity needed for stability is very dissipative.

- Free surface
  A free surface in SPH is simply the transition between particles and nothing. Large

variations in the free surface, or intersecting surfaces are no problem at all. Overtopping of waves is found from the Navier-Stokes equations without further assumptions.

- Boundaries
  Closed boundaries are created with a line of fixed boundary particles with a boundary force and ghost particles over the boundary. Closed boundaries sometimes disturb the flow nearby. Open boundaries are difficult to apply in SPH. For the case of a sharp weir an inflow boundary was created which gave only a small disturbance on the flow. This inflow boundary can only handle a uniform velocity in one direction. Periodic boundaries can be used without problems.

- Calculation time
  A small time step is needed for stability and finding all the interaction pairs every time step is computational expensive. The calculation times therefore are considerable. The usability of SPH is restricted to local and short phenomena.

- Future
  The simulations in chapter 6 show that SPH can give accurate results for many problems in hydraulic engineering. SPH can be used in situations where other numerical methods fail. The possibilities of SPH in hydraulic engineering are therefore numerous and with some improvements it will be used more and more.

7.3 Recommendations

- Reduction of calculation time
  The calculation time for simulations with large numbers of particles can be reduced with a more sophisticated interaction find algorithm. Parallelisation of the code can also decrease the calculation times significantly.

- Expansion of SPH to 3D
  With a faster code and increasing computer power over time, soon the calculation time for 3D simulations with satisfactory resolution will be acceptable.

- Improvement of the viscosity model
  The currently used artificial viscosity approach is too dissipative. A less dissipative viscosity model, which keeps the simulation stable, would give better results in turbulent situations. Some papers are already published about a combination of SPH with the concept of large eddy simulation (LES). The particle movement from the Navier-Stokes equations is used to capture the large scale motion, and a sub-particle-scale (SPS) turbulence model is used to calculate the influence of turbulence from smaller length scales.

- Improvement of the formulation of boundaries
  An improvement in the way closed and open boundaries are modelled is welcome. Ro-

bust techniques to model open boundaries without negative boundary effects would make SPH applicable in more situations.

- Make the fluid incompressible
  The compressibility of water is exaggerated in SPH to get a time derivative of the density and to get a relation between density and pressure. An approach closer to reality would be to model water as an incompressible fluid, and to solve an implicit Poisson equation for the pressure.

- Staggering of information over different types of particles
  In the very structured 2D horizontal viscosity tests a disturbing checkerboard mode entered the pressure of the particles. SPH is normally used in very dynamic situations, then the disturbing checkerboard mode does not show up. A fundamental correct solution would be to introduce two types of particles: particles which carry information about pressure and particles which carry information about velocity. From grid based methods it is known this prevents checkerboard modes and it does not reduce the accuracy. However this will not be easy to implement in a particle method.

Bibliography

Batchelor, G. (1974). *An introduction to Fluid Mechanics*. Cambridge University Press.

Battjes, J. (2001). *Korte Golven*. Lecture notes ct4320 TU Delft. 6.6.2, 6.6.2

Bonet, J. and S. Kulasegaram (2000). Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations. *International journal for numerical methods in engineering 47*, 1189–1214.

Bonet, J., S. Kulasegaram, M. Rodriquez-Paz, and M. Profit (2004). Variatonal formulation for the smooth particle hydrodynamics (sph) simulation of fluid and solid problems. *Computational methods applied mechanical engineering 193*, 1245–1256.

Bonet, J. and T.-S. Lok (1999). Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computational methods applied mechanical engineering 180*, 97–115. 4.5, 6.3, 6.4

Gómez-Gesteira, M. and R. Dalrymple (2004). Using a three-dimensional SPH method for wave impact on a tall structure. *Journal of Waterway, Port, Coastal, and Ocean Engineering 130*(2), 63–69. 2.5

Harlow, F. (1963). The particle-in-cell method for numerical solution of problems in fluid dynamics. *Experimental arithmetic, high-speed computations and mathematics 15*, 269.

Harlow, F. and J. Welch (1965). Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids 8*(12), 2182–2189. 6.3

Hirt, C. and B. Nichols (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics 39*, 201. 2.4

Liu, G. and M. Liu (2003). *Smoothed Particle Hydrodynamics*. World Scientific Publishing Co. Pte. Ltd. 3.3.1, 3.3.2, 5

Martin, J. and W. Moyce (1952). An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical transactions of the Royal Society of London 244*(882), 312–324. 6.3, 6.3

Monaghan, J. (1992). Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 543–574. 4.2.1, 6.3

Monaghan, J. (1994). Simulating free surface flows with SPH. *Journal of computational physics 110*, 399–406. 3.3.2

Monaghan, J. (2000). SPH without a tensile instability. *Journal of computational physics 159*, 290–311.

Monaghan, J. (2005). Smoothed particle hydrodynamics. *Reports on progress in physics 68*, 1703–1759.

Monaghan, J., A. Kos, and N. Issa (2003). Fluid motion generated by impact. *Journal of Waterway, Port, Coastal, and Ocean Engineering 129*(6), 250–259. 4.2.1

Rogers, B. and R. Dalrymple (2004). SPH modeling of breaking waves. *Coastal Engineering*, 415–426. 2.5

Wesseling, P. (2002). *Elements of computational fluid dynamics*. Lecture notes wi4011 TU Delft.

Appendix A

Listings of 'SPH 2D open' computer code

SPH_2D.f

```
      program SPH

      implicit none

      include 'param.txt'

      integer maxtimestep, itime, d, m, i, printstep, screenstep, niac
      double precision dt, rho_min(maxn), v_min(dim,maxn), drhodt(maxn),
     & dvdt(dim,maxn), av(dim,maxn), t1, t2, xl, yl

      dt = 0.0005
      maxtimestep = 10
      printstep = 1
      screenstep = 1

      call cpu_time(t1)

      call input (x, v, mass, rho, p, u, itype, hsml, ntotal,
     & nvirt, xl, yl)

          do itime=1,maxtimestep                    ! save rho&v last timestep
          if (itime.ne.1) then                      ! predict rho&v half timestep
            do i=1,ntotal
              rho_min(i)=rho(i)
              rho(i)=rho(i)+0.5*dt*drhodt(i)
              do d=1,dim
                v_min(d,i)=v(d,i)
                v(d,i)=v(d,i)+0.5*dt*dvdt(d,i)
                vXSPH(d,i)=vXSPH(d,i)+0.5*dt*dvdt(d,i)
              enddo
            enddo
          endif
        if (itime.eq.1) then
          call derivatives (x, v, v, mass, rho, p, u, hsml, ntotal,
     &    nvirt, nvII, drhodt, dvdt, av, niac, itime, dt, rho_min, v_min)
        else
          call derivatives (x, v, vXSPH, mass, rho, p, u, hsml, ntotal,
     &    nvirt, nvII, drhodt, dvdt, av, niac, itime, dt, rho_min,
```

```
     &       v_min)
40         endif

           if (itime.eq.1) then                    ! first timestep different integration
             do i=1,ntotal
               rho(i)=rho(i)+0.5*dt*drhodt(i)
45             do d=1,dim
                 vXSPH(d,i)=v(d,i)+0.5*dt*dvdt(d,i)+av(d,i)
                 v(d,i)=v(d,i)+0.5*dt*dvdt(d,i)
                 x(d,i)=x(d,i)+dt*vXSPH(d,i)
               enddo
50           enddo
           else                                    ! normal time-integration
             do i=1,ntotal
               rho(i)=rho_min(i)+dt*drhodt(i)
               do d=1,dim
55               vXSPH(d,i)=v_min(d,i)+dt*dvdt(d,i)+av(d,i)
                 v(d,i)=v_min(d,i)+dt*dvdt(d,i)
                 x(d,i)=x(d,i)+dt*vXSPH(d,i)
               enddo
             enddo
60         endif

           if(mod(itime,printstep).eq.0) then
             call output(x, v, vXSPH, mass, rho, p, u, itype, hsml,
     &     ntotal, nvirt, nvII, itime)
65         endif


           if(mod(itime,screenstep).eq.0) then
             write(*,*)'  ****************************************'
70           write(*,*)'    Timestep: ', itime, ' of ', maxtimestep
             write(*,*)'    Interactionpairs: ', niac
             write(*,*)'  ****************************************'
           endif
         enddo
75

         call cpu_time(t2)
         write(*,'(a,f9.2,a)')'    CPU time: ', t2-t1, ' s. '

         end program
```

input.f

```
      subroutine input (x, v, mass, rho, p, u, itype, hsml, ntotal,
     &  nvirt, xl, yl)

c       ------------------------------------------------------------------------
c       Subroutine to create initial fluid particles. *** In 'SPH closed' in this
c       subroutine also the boundary particles are created. ***
c
c       Variables used in call:
c       x       :particle position in 2 dimensions                      [out]
c       v       :particle velocity in 2 dimensions                      [out]
c       mass    :particle mass                                          [out]
c       rho     :particle density                                       [out]
c       p       :particle pressure                                      [out]
c       u       :particle energy (not used)                             [out]
c       itype   :number indicating fluid or boundary (not used)         [out]
c       hsml    :particle smoothing length                              [out]
c       ntotal  :number of fluid particles                              [out]
c       nvirt   :number of boundary particles                           [out]
c       xl      :max horizontal position of the fluid                   [out]
c       yl      :max vertical position of a the fluid                   [out]
c       ------------------------------------------------------------------------

        implicit none

        include 'param.txt'

        integer nx, ny, ix, iy, i, j, d, im, mx, my
        double precision xl, yl, dx, space

c       ------------------------------------------------------------------------
c       Load initial fluid particle input from file

        if(input_file) then
        open(1,file="data/ini_xv.dat")
        open(2,file="data/ini_state.dat")
        open(3,file="data/ini_other.dat")

        write(*,*)'  -----------------------------------------------'
        write(*,*)'     Loading initial particle configuration...'
          read (1,*) ntotal
        write(*,*)'     Total number of particles : ', ntotal
        write(*,*)'  -----------------------------------------------'
        do i = 1, ntotal
          read(1,*)im, (x(d, i),d = 1, dim), (v(d, i),d = 1, dim)
          read(2,*)im, mass(i), rho(i), p(i), u(i)
          read(3,*)im, itype(i), hsml(i)
        enddo

       else

c       ------------------------------------------------------------------------
c       Generate initial fluid particles in this subroutine

        open(1,file="data/ini_xv.dat",status='replace')
        open(2,file="data/ini_state.dat",status='replace')
        open(3,file="data/ini_other.dat",status='replace')

          nx=200        ! outer number of particles in staggered way  o   o   o
          ny=12         ! this example is 2x3                           o   o
```

```
60            space=0.025   ! size is 2dx 1dy                                o   o   o
              i=0
              ntotal=nx*ny !+(nx-1)*(ny-1)
c         Outside grid
              do ix= 1,nx
65                  do iy= 1, ny
                            i=i+1
                            x(1,i)=-1.*space+ix*space
                            x(2,i)=0.*space+iy*space
                    enddo
70            enddo
c         Second staggered grid inside first one
c             do ix=1,nx-1
c               do iy= 1,ny-1
c                   i=i+1
75  c                       x(1,i)=0.5*space+ix*space
c                           x(2,i)=0.5*space+iy*space
c                   enddo
c             enddo

80            do i=1, ntotal
                    v(1,i)=0.5
                    v(2,i)=0.0
                    rho(i)=1000*(1+(9810/6.914e4*(x(2,ny)-x(2,i))))**(1./7)
                    mass(i)=1000*space*space
85                  p(i)=0.
                    u(i)=0.
                    hsml(i)=0.035
                    itype(i)=2
              enddo
90

c         Write just derived particle info to file

              write(1,*) ntotal
95            do i = 1, ntotal
              write(1,1001) i, (x(d, i),d = 1, dim), (v(d, i),d = 1, dim)
              write(2,1002) i, mass(i), rho(i), p(i), u(i)
              write(3,1003) i, itype(i), hsml(i)
              enddo
100   1001    format(1x, I5, 6(2x, e14.8))
      1002    format(1x, I5, 7(2x, e14.8))
      1003    format(1x, I5, 2x, I2, 2x, e14.8)
              write(*,*)'  ************************************************'
              write(*,*)'      Initial particle configuration generated   '
105           write(*,*)'      Total number of particles   ', ntotal
              write(*,*)'  ************************************************'

            endif

110         close(1)
            close(2)
            close(3)

              end subroutine
```

`derivatives.f`

```
      subroutine derivatives (x, v, vXSPH, mass, rho, p, u, hsml, ntotal
     &  ,nvirt, nvII, drhodt, dvdt, av, niac, itime, dt, rho_min, v_min)

c     ------------------------------------------------------------------------
c     Subroutine to solve the SPH equations every timestep.
c     All interactions are calculated per interactionpair and then added to
c     both particles belonging to that interactionpair.
c
c     Variables used in call:
c     x       :particle position in 2 dimensions                    [in]
c     v       :particle velocity in 2 dimensions                    [in]
c     vXSPH   :particle XSPH velocity in 2 dimensions               [in]
c     mass    :particle mass                                        [in]
c     rho     :particle density                                     [in]
c     p       :particle pressure                                    [out]
c     u       :particle energy (not used)                           [in]
c     hsml    :particle smoothing length                            [in]
c     ntotal  :number of fluid particles                            [in]
c     nvirt   :number of boundary particles                         [in]
c     nvII    :number of ghost particles                            [out]
c     drhodt  :time derivative of density                           [out]
c     dvdt    :time derivative of velocity in 2 dimensions          [out]
c     av      :particle average velocity contribution of XSPH       [out]
c     niac    :total number of interactions                         [out]
c     itime   :current timestep number                              [in]
c     dt      :timestep                                             [in]
c     rho_min :particle density from previous timestep              [out]
c     v_min   :particle velocity in 2 dim. from previous timestep   [out]
c     ------------------------------------------------------------------------

      implicit none

      include 'param.txt'

      integer i, j, k, d, niac, i_pair(max_int), j_pair(max_int), itime,
     &  ii, i_inflow(maxn), inflowparts, ncheck

      double precision drhodt(max_int), dvdt(dim,max_int), av(dim,maxn),
     &  r, r2, dx(dim), rij2(max_int), q, alpha_d, W(max_int),
     &  dWdx(dim,max_int), vijdWdx, c, B, ppart, mrho, epsilon,
     &  avgpart, h, f, RR, alpha, beta, eta2, artvisc, g, dt, vijrij,
     &  mu, xl, vl, r0, DD, pb, xshore, yshore, rho_min(maxn),
     &  v_min(dim,maxn), sumdvdt, sumav

      logical check1

      save inflowparts, i_inflow


c     After moving particles inflow and outflow
c     *** Not used in 'SPH closed' ***
      ncheck=ntotal
      if (itime.gt.1) then
        call inoutflow (x, v, vXSPH, mass, rho, p, u, hsml,
     &   ntotal, i_inflow, inflowparts, rho_min, v_min)
      endif
c     With inflow/outflow boundaries call inputbp every timestep
c     *** Not used in 'SPH closed' ***
      call inputbp(x, v, mass, rho, p, u, itype, hsml, ntotal,
```

```
60       &  nvirt, xl, vl, itime, dt)

c        Track particles within space from inflow boundary
c        *** Not used in 'SPH closed' ***
         k=0
65       do i=1,ntotal
           if (x(1,i).le. 0.0250000) then
           k=k+1
           i_inflow(k)=i
           endif
70       enddo
         inflowparts=k

         do i=ntotal+1,ntotal+nvirt  !boundary parts also need vXSPH
           do d=1,dim
75           vXSPH(d,i)=v(d,i)
           enddo
         enddo

c        Create virtII ghost particles within 2hsml from boundary
80
         ii=ntotal+nvirt

c        Left boundary
         do k=1,inflowparts
85         do j=1,2
             ii=ii+1
                 i=i_inflow(k)
                 x(1,ii)=-j*0.025+x(1,i)
                 x(2,ii)=x(2,i)
90               v(1,ii)=v(1,i)
                 v(2,ii)=v(2,i)
                 vXSPH(1,ii)=vXSPH(1,i)
                 vXSPH(2,ii)=vXSPH(2,i)
                 mass(ii)=mass(i)
95               rho(ii)=rho(i)
                 p(ii)=p(i)
                 u(ii)=u(i)
           hsml(ii)=hsml(i)
           enddo
100      enddo
c        Lower boundary
         do i=1,ntotal
           if (x(2,i).le.2*hsml(i).and.x(1,i).le.2) then
             ii=ii+1
105              x(1,ii)=x(1,i)
                 x(2,ii)=-x(2,i)
                 v(1,ii)=v(1,i)
                 v(2,ii)=-v(2,i)
                 vXSPH(1,ii)=vXSPH(1,i)
110              vXSPH(2,ii)=-vXSPH(2,i)
                 mass(ii)=mass(i)
                 rho(ii)=rho(i)
                 p(ii)=p(i)
                 u(ii)=u(i)
115        hsml(ii)=hsml(i)
           endif
c        Right boundary
           if (x(1,i).ge.1.93.and.x(1,i).lt.2.and.x(2,i).le.0.2) then
             ii=ii+1
```

```
120                 x(1,ii)=4-x(1,i)
                    x(2,ii)=x(2,i)
                    v(1,ii)=-v(1,i)
                    v(2,ii)=v(2,i)
                    vXSPH(1,ii)=-vXSPH(1,i)
125                 vXSPH(2,ii)=vXSPH(2,i)
                    mass(ii)=mass(i)
                    rho(ii)=rho(i)
                    p(ii)=p(i)
                    u(ii)=u(i)
130                 hsml(ii)=hsml(i)
              endif
            enddo
            nvII=ii-ntotal-nvirt

135 c       Initializing: make derivatives zero at start, must be after inoutflow.f
            do i=1,ntotal+nvirt+nvII
              drhodt(i)=0.0
              do d=1,dim
                dvdt(d,i)=0.0
140             av(d,i)=0.0
              enddo
            enddo

    c       Interaction + kernel
145         niac=0
            do i=1,ntotal                    ! first part interaction is real part
              do j=i+1,ntotal+nvirt+nvII     ! interaction only j>i
                check1=x(1,i).ge.2.and.j.gt.ntotal+nvirt
    c           *** Specific for this simulation ***
150 c           The ghost particles outside the weir may not have interaction with the
    c           fluid jet flowing over the weir, check1 makes sure this is the case.
                  if(.not.check1) then
                    do d=1,dim
                      dx(d)=x(d,i)-x(d,j)
155                 enddo
                    r2=dx(1)**2+dx(2)**2
                    h=(hsml(i)+hsml(j))/2
                    if (r2.le.4.*h*h) then
                      niac=niac+1
160                   i_pair(niac)=i
                      j_pair(niac)=j
                      rij2(niac)=r2
    c       Kernel is Piecewise cubic spline (Monaghan 1985)
                        r=sqrt(r2)
165                     q=r/h
                        alpha_d=10./(7.*pi*h*h)
                        if (q.ge.0.and.q.lt.1.0) then
                          W(niac)=alpha_d*(1-1.5*q*q+0.75*q**3)
                          do d=1,dim
170                             dWdx(d,niac)=alpha_d*(-3+2.25*q)/h**2 * dx(d)
                              enddo
                        else if (q.ge.1.0.and.q.le.2.0) then
                          W(niac)=alpha_d*0.25*(2-q)**3
                          do d=1,dim
175                         dWdx(d,niac)=-alpha_d*0.75*(2-q)**2*dx(d)/(h*r)
                          enddo
                        else
                          W(niac)=0
                          do d=1,dim
```

```
                        dWdx(d,niac)=0.
                      enddo
                    endif
                  endif
                endif
            enddo
          enddo

c         Density
          do k=1,niac
            i=i_pair(k)
            j=j_pair(k)
            do d=1,dim
              vijdWdx=(vXSPH(d,i)-vXSPH(d,j))*dWdx(d,k)
              drhodt(i)=drhodt(i)+mass(j)*vijdWdx
              if(j.le.ntotal) drhodt(j)=drhodt(j)+mass(i)*vijdWdx ! double change in sign=OK
            enddo
          enddo

c         Equation of State
          do i=1,ntotal+nvirt
            c=22.
            B=6.914e4
            p(i)=B*((rho(i)/1000.)**7-1)
          enddo

c         Momentum equation
          do k=1,niac
            i=i_pair(k)
            j=j_pair(k)
            RR=0.
            f=0.
            ppart=(p(i)/rho(i)**2+p(j)/rho(j)**2)
            if (virt_pres) then          !virt_pres for removing tensile instability
              f=W(k)/(0.5*alpha_d)       !W(dp)=W(q=0.71)/alpha_d=0.5
                if (p(i).lt.0) RR=-0.2*p(i)/(rho(i)*rho(i))
                if (p(j).lt.0) RR=RR-0.2*p(j)/(rho(j)*rho(j))
                if (p(i).gt.0.and.p(j).gt.0) RR=0.01*ppart
            endif
            do d=1,dim
               dvdt(d,i)=dvdt(d,i)-mass(j)*(ppart+RR*f**4)*dWdx(d,k)
                   if(j.le.ntotal) dvdt(d,j)=dvdt(d,j)+mass(i)*
     &                                    (ppart+RR*f**4)*dWdx(d,k)
            enddo
          enddo

c         Artificial viscosity
          if (art_visc) then
            alpha=0.01
            do k=1,niac
              i=i_pair(k)
              j=j_pair(k)
              if(j.le.ntotal) then
                vijrij=0.
                mrho=(rho(i)+rho(j))/2
                    do d=1,dim
                      vijrij=vijrij+(v(d,i)-v(d,j))*(x(d,i)-x(d,j))
                enddo
                if (vijrij<0) then
                  h=(hsml(i)+hsml(j))/2
```

```
240                   eta2=0.01*h*h
                      mu=h*vijrij/(rij2(k)+eta2)
                      artvisc=-alpha*c*mu/mrho
                      do d=1,dim
                        dvdt(d,i)=dvdt(d,i)-mass(j)*artvisc*dWdx(d,k)
245                         if(j.le.ntotal)  dvdt(d,j)=dvdt(d,j)+mass(i)*
     &                                          artvisc*dWdx(d,k)
                      enddo
                    endif
                  endif
250             enddo
              endif

c             Gravity (only for real particles)
              g=9.81
255           do i=1,ntotal
                dvdt(dim,i)=dvdt(dim,i)-g
              enddo

c             Lennard Jones boundary force
260           r0=0.025
              DD=0.1
              do j=ntotal+1,ntotal+nvirt
                do i=1,ntotal ! i=fluid particle
                  do d=1,dim
265                 dx(d)=x(d,i)-x(d,j)
                  enddo
                  r2=dx(1)**2+dx(2)**2
                  if(r2<r0*r0) then
                     r=sqrt(r2)
270                      pb=DD/r2*((r0/r)**12-(r0/r)**4)
                         do d=1,dim
                          dvdt(d,i)=dvdt(d,i)+pb*dx(d)
                     enddo
                  endif
275             enddo
              enddo

c             Avg velocity (XSPH)
              if (avg_vel) then
280             epsilon=0.5
                do k=1,niac
                  i=i_pair(k)
                  j=j_pair(k)
                      if(j.le.ntotal) then !XSPH only with fluid particles not with boundary
285                     mrho=(rho(i)+rho(j))/2
                    do d=1,dim
                      avgpart=epsilon*(vXSPH(d,j)-vXSPH(d,i))/mrho*W(k)
                          av(d,i)=av(d,i)+mass(j)*avgpart
                          av(d,j)=av(d,j)-mass(i)*avgpart
290                 enddo
                  endif
                enddo
              endif

295 c         *** Specific for this simulation ***
    c         The flow after the weir may not pull.
              do k=1,niac
                i=i_pair(k)
                j=j_pair(k)
```

```
300             if(x(1,i).ge.2.or.x(1,j).ge.2) then
                  av(2,i)=0.
                  av(2,j)=0.
                endif
              enddo
305
      c       *** Specific for this simulation ***
      c       Get average inflowvelocity
              sumdvdt=0.
              sumav=0.
310           do k=1,inflowparts
                i=i_inflow(k)
                sumdvdt=sumdvdt+dvdt(1,i)
                sumav=sumav+av(1,i)
              enddo
315
      c       *** Specific for this simulation ***
      c       Particles within hsml from inflow all have the same inflow-velocity
      c       *** Not used in 'SPH closed' ***
              do k=1,inflowparts
320             i=i_inflow(k)
                dvdt(2,i)=0.
                av(2,i)=0.
                dvdt(1,i)=sumdvdt/inflowparts
                av(1,i)=sumav/inflowparts
325   c         A particle within hsml from inflow is not allowed to flow back
                if (v(1,i).le.0.and.dvdt(1,i).le.0.) dvdt(1,i)=0.
                if (v(1,i).le.0.and.av(1,i).le.0.) av(1,i)=0.
                drhodt(i)=0. !Combination of depth with p (&rho) is fixated
              enddo
330
      c     Generate textfile with useful info about simulation
              if (itime.eq.1) then
                call parameterfile(ntotal, nvirt, hsml, mass, c, B, g,
            &     alpha, epsilon, dt)
335           endif

              end subroutine
```

inoutflow.f

```
      subroutine inoutflow (x, v, vXSPH, mass, rho, p, u, hsml,
     &   ntotal, i_inflow, inflowparts, rho_min, v_min)


c     ------------------------------------------------------------------------
c     Subroutine to generate fluid particles at inflow and remove particles
c     at outflow. Only vertical uniform velocity profile at inflow is possible.
c
c     Variables used in call:
c     x       :particle position in 2 dimensions                    [in/out]
c     v       :particle velocity in 2 dimensions                    [in/out]
c     vXSPH   :particle XSPH velocity in 2 dimensions               [in/out]
c     mass    :particle mass                                        [in/out]
c     rho     :particle density                                     [in/out]
c     p       :particle pressure                                    [in/out]
c     u       :particle energy (not used)                           [in/out]
c     hsml    :particle smoothing length                            [in/out]
c     ntotal  :number of fluid particles                            [in/out]
c     i_inflow:array with particles within hsml from inflow         [in]
c     inflowparts:total number of particles within hsml from inflow [in]
c     rho_min :particle density from previous timestep              [in/out]
c     v_min   :particle velocity in 2 dim. from previous timestep   [in/out]
c     ------------------------------------------------------------------------

      implicit none

      include 'param.txt'

      integer i, j, k, d, ii, i_inflow(maxn), inflowparts, jj

      double precision rij2, x2save, psave, rhosave, rho_min(maxn),
     &  v_min(dim,maxn), x2move, inputmass, inputvol

c     Left inflow
      ii=ntotal
      inputvol=0
      inputmass=0
      do k=1,inflowparts
        i=i_inflow(k)
        if (x(1,i).gt.0.0250000) then
          ii=ii+1
          x(1,ii)=x(1,i)-0.025
          x(2,ii)=x(2,i)
          v(1,ii)=v(1,i)
          v(2,ii)=0.
          vXSPH(1,ii)=vXSPH(1,i)
          vXSPH(2,ii)=0.
          mass(ii)=mass(i)
          rho(ii)=rho(i)
          p(ii)=p(i)
          u(ii)=u(i)
          hsml(ii)=hsml(i)
          rho_min(ii)=rho(i)
          v_min(1,ii)=v(1,i)
          v_min(2,ii)=0.
          ! Track input discharge
          inputmass=inputmass+mass(ii)
          inputvol=inputvol+mass(ii)/rho(ii)
        endif
```

```
60
          enddo
          if (inputmass.ne.0) write(*,'(a,f6.2,a,f6.4,a)') 'input mass: ',
      &   inputmass, ' kg/m;  input vol: ', inputvol, ' m2'

65        ntotal=ii

c         Right outflow
          do i=1,ntotal
            if(x(2,i).lt.0.) then
70            x(1,i)=x(1,ntotal)
              x(2,i)=x(2,ntotal)
              v(1,i)=v(1,ntotal)
              v(2,i)=v(2,ntotal)
              vXSPH(1,i)=vXSPH(1,ntotal)
75            vXSPH(2,i)=vXSPH(2,ntotal)
              mass(i)=mass(ntotal)
              rho(i)=rho(ntotal)
              p(i)=p(ntotal)
              u(i)=u(ntotal)
80            hsml(i)=hsml(ntotal)
              rho_min(i)=rho_min(ntotal)
              v_min(1,i)=v_min(1,ntotal)
              v_min(2,i)=v_min(2,ntotal)
              ntotal=ntotal-1
85          endif
          enddo

          end subroutine
```

inputbp.f

```
      subroutine inputbp (x, v, mass, rho, p, u, itype, hsml, ntotal,
     &    nvirt, xl, vl, itime, dt)

c     ------------------------------------------------------------------------
c     Subroutine to create boundary particles. *** In 'SPH closed' this
c     subroutine is part of input.f ***
c
c     Variables used in call:
c     x       :particle position in 2 dimensions                      [in/out]
c     v       :particle velocity in 2 dimensions                      [in/out]
c     mass    :particle mass                                          [in/out]
c     rho     :particle density                                       [in/out]
c     p       :particle pressure                                      [in/out]
c     u       :particle energy (not used)                             [in/out]
c     itype   :number indicating fluid or boundary (not used)         [in/out]
c     hsml    :particle smoothing length                              [in/out]
c     ntotal  :number of fluid particles                              [in]
c     nvirt   :number of boundary particles                           [out]
c     xl      :horizontal position of a moving boundary               [out]
c     vl      :horizontal velocity of a moving boundary               [out]
c     itime   :current timestep number                                [in]
c     dt      :timestep                                               [in]
c     ------------------------------------------------------------------------

      implicit none

      include 'param.txt'

      integer nx, ny, ix, iy, itime, i, j, d, im, mx, my
      double precision xl, vl, dx, space, dt

c     ------------------------------------------------------------------------
c     Load boundary particle input from file
      if (vp_file) then

      open(1,file="data/xv_vp.dat")
      open(2,file="data/state_vp.dat")
      open(3,file="data/other_vp.dat")
      read(1,*) nvirt
      do j = 1, nvirt
        i = ntotal + j
        read(1,*)im, (x(d, i),d = 1, dim), (v(d, i),d = 1, dim)
        read(2,*)im, mass(i), rho(i), p(i), u(i)
        read(3,*)im, itype(i), hsml(i)
      enddo
      close(1)
      close(2)
      close(3)

c     ------------------------------------------------------------------------
c     Generate boundary particles in this subroutine
      else

      nvirt = 0
      dx = 0.025

c   Boundary particles on the Lower side
      do i = 1, 81
            nvirt = nvirt + 1
```

```
60              x(1, ntotal + nvirt) = (i-1)*dx
          x(2, ntotal + nvirt) = 0.
          v(1, ntotal + nvirt) = 0.
              v(2, ntotal + nvirt) = 0.
        enddo
65
    c   Boundary particles on the Left side (wave maker)
    c     do i = 1, 50
    c       nvirt = nvirt + 1
    c           x(1, ntotal + nvirt) = -1.25*cos(2*itime*dt)+1.25
70  c       x(2, ntotal + nvirt) = i*dx
    c       v(1, ntotal + nvirt) = 1.25*2*sin(2*itime*dt)
    c           v(2, ntotal + nvirt) = 0.
    c     enddo
    c         xl=-1.25*cos(2*itime*dt)+1.25
75  c         vl=1.25*2*sin(2*itime*dt)

    c   Boundary particles on the Right side
        do i = 1, 7
          nvirt = nvirt + 1
80            x(1, ntotal + nvirt) = 2
          x(2, ntotal + nvirt) = i*dx
          v(1, ntotal + nvirt) = 0.
              v(2, ntotal + nvirt) = 0.
        enddo
85
    c       Give physical properties
            do i = 1, nvirt
              rho (ntotal + i) = 1000.
              mass(ntotal + i) = rho (ntotal + i) * dx * dx
90            p(ntotal + i) = 0.
              u(ntotal + i) = 357.1
              itype(ntotal + i) = -2
              hsml(ntotal + i) = 0.035
        enddo
95      endif


    c       Write data virt part to file one time
          if(itime == 1) then
100 1001  format(1x, I5, 6(2x, e14.8))
    1002  format(1x, I5, 7(2x, e14.8))
    1003  format(1x, I5, 2x, I2, 2x, e14.8)
            open(1,file="data/xv_vp.dat")
            open(2,file="data/state_vp.dat")
105         open(3,file="data/other_vp.dat")
            write(1,*) nvirt
            do i = ntotal + 1, ntotal + nvirt
              write(1,1001) i, (x(d, i), d=1,dim), (v(d, i), d = 1, dim)
              write(2,1002) i, mass(i), rho(i), p(i), u(i)
110           write(3,1003) i, itype(i), hsml(i)
            enddo
            close(1)
            close(2)
            close(3)
115       endif

          end subroutine
```

output.f

```
      subroutine output(x, v, vXSPH, mass, rho, p, u, itype, hsml,
     &     ntotal, nvirt, nvII, itime)

          implicit none

c         ------------------------------------------------------------------------
c         Subroutine to write date to output file.
c
c         Variables used in call:
c         x        :particle position in 2 dimensions                      [in]
c         v        :particle velocity in 2 dimensions                      [in]
c         vXSPH    :particle XSPH velocity in 2 dimensions                 [in]
c         mass     :particle mass                                          [in]
c         rho      :particle density                                       [in]
c         p        :particle pressure                                      [in]
c         u        :particle energy (not used)                             [in]
c         itype    :number indicating fluid or boundary (not used)         [in]
c         hsml     :particle smoothing length                              [in]
c         ntotal   :number of fluid particles                              [in]
c         nvirt    :number of boundary particles                           [in]
c         nvII     :number of ghost particles                              [in]
c         itime    :current timestep number                                [in]
c         ------------------------------------------------------------------------

          include 'param.txt'

          integer d, i, itime
          character*30 filename_xv, filename_state, filename_other

          write (filename_xv,'(a, i8.8, a)') 'data/f_xv', itime, '.dat'

       open(1,file=filename_xv, status='replace')

       do i = 1, ntotal+nvirt+nvII
         write(1,1001) i, (x(d, i), d=1,dim), (v(d, i), d = 1, dim),
     &     rho(i), p(i), mass(i)
         enddo

1001   format(1x, I6, 7(2x, e14.8))

         close(1)

           end subroutine
```

parameterfile.f

```
      subroutine parameterfile(ntotal, nvirt, hsml, mass, c, B, g,
     &    alpha, epsilon, dt)

      implicit none

c     ------------------------------------------------------------------------
c     Subroutine to write textfile with info about simulation.
c
c     Variables used in call:
c     ntotal  :number of fluid particles                              [in]
c     nvirt   :number of boundary particles                           [in]
c     hsml    :particle smoothing length                              [in]
c     mass    :particle mass                                          [in]
c     c       :used velocity of sound                                 [in]
c     B       :used factor in eq. of state for pressure               [in]
c     g       :gravity constant                                       [in]
c     alpha   :constant in artificial viscosity                       [in]
c     epsilong:constant in calculation of XSPH                        [in]
c     dt      :used timestep                                          [in]
c     ------------------------------------------------------------------------

      include 'param.txt'

      double precision c, B, g, alpha, epsilon, dt
      character*30 filename

      write (filename,'(a)') 'data/Simulationinfo.txt'

    open(1,file=filename, status='replace')

      write(1,*) '-------------------------------------------------'
      write(1,*) '      Smoothed Particle Hydrodynamics 2D model'
      write(1,*) '    boundary with LJ force and virt II particles'
      write(1,*) '            Information about simulation'
      write(1,*) '-------------------------------------------------'

      write(1,*)
      write(1,'(a,i7)') 'Number of fluid particles:    ', ntotal
      write(1,'(a,i7)') 'Number of boundary particles: ', nvirt
      write(1,'(a,f8.5,a)') 'Timestep: ', dt, ' s'
      write(1,*)
      write(1,*)
      write(1,'(a)') 'Processes included:'
      if (art_visc) then
        write(1,'(a,l3)') '   Artificial viscosity:', art_visc
     write(1,'(a,f5.3)') '     * alpha = ',alpha
      else
        write(1,'(a,l3)') '   Artificial viscosity:', art_visc
      endif

      if (avg_vel) then
        write(1,'(a,l3)') '   Average velocity:', avg_vel
     write(1,'(a,f5.3)') '     * epsilon = ',epsilon
      else
        write(1,'(a,l3)') '   Average velocity:', avg_vel
      endif
      write(1,'(a,l3)') '   Prevention tensile instability:', virt_pres

      write(1,*)
```

```
60          write(1,*)
            write(1,'(a)') 'Particle information:'
            write(1,'(a,f7.4,a)') '   Hsml fluid particles: ', hsml(1),' m'
            write(1,'(a,f7.4,a)') '   Hsml boundary particles: ',
        &   hsml(ntotal+1),' m'
65          write(1,'(a,f7.4,a)') '   Mass fluid particles: ', mass(1),' kg/m'
            write(1,'(a,f7.4,a)') '   Mass boundary particles: ',
        &   mass(ntotal+1),' kg/m'

            write(1,*)
70          write(1,*)
            write(1,'(a)') 'Parameters:'
            write(1,'(a,f5.2,a)') '   Gravity, g = ',g, ' m/s^2'
            write(1,'(a,f6.2,a)') '   Speed of sound, c = ',c, ' m/s'
            write(1,'(a,e10.4,a)') '   Compressionfactor B = c^2*rho/7 = ',B,
75      &    ' kg/(ms^2)'
            write(1,*)
            write(1,*)
            write(1,*) '-------------------------------------------------'
            write(1,*) '  programmed by:   L de Wit'
80          write(1,*) '                   Graduation project'
            write(1,*) '                   Section Fluidmechanics '
            write(1,*) '                   Faculty Civil Engineering'
            write(1,*) '                   TU Delft'
            write(1,*) '                   September 2005'
85          write(1,*) '-------------------------------------------------'

            end subroutine
```

param.txt

```
c-------------------------------------------------------------
c     Including file for parameters and constants used
c     in all SPH_2D files.
c-------------------------------------------------------------

        integer maxn, dim, max_int

        parameter (maxn=12000, dim=2, max_int= 100*maxn)

        integer itype(maxn), ntotal, nvirt, nvII
        double precision x(dim,maxn), v(dim,maxn), vXSPH(dim,maxn),
     & mass(maxn), p(maxn), u(maxn), hsml(maxn), rho(maxn), pi

        parameter ( pi = 3.14159265358979323846 )

        logical virt_pres, art_visc, avg_vel, input_file, vp_file

        parameter ( virt_pres=.true.)  !virt_pres to prevent tensile instability
        parameter ( art_visc=.true.)   !art_visc to simulate viscosity
        parameter ( avg_vel=.true.)    !avg_vel for XSPH variant.
        parameter ( input_file=.true.) !input_file = T input fluid particles from files
        parameter ( vp_file=.false.)   !vp_file = T input boundary particles from files
```

Appendix B

List of Symbols

LATIN		
B	Constant used in the equation of state (eq. 3.7) $B = c^2\rho_0/\gamma$	N/m^2
c	Velocity of sound in water	m/s
D	Constant used in the boundary force	m^2/s
d	Water depth	m
$\mathbf{f}_b$	Vector with acceleration due to a body force	m/s^2
$\mathbf{f}_i$	Vector with acceleration due to a body force on particle i	m/s^2
$f(\mathbf{x})$	Continuous function	-
$\langle f(\mathbf{x})\rangle$	Approximation of function f at $\mathbf{x}$	-
g	Gravitational constant $g = 9.81$ m/s^2	m/s^2
h	Smoothing length	m
$_{i,j}$	Subscripts indicating particle index	-
k	Wave number	rad/m
m	Mass, in 2D mass/length	kg/m
^{n}	Superscript indicating time step	-
p	Pressure	N/m^2
q	$q = r_{ij}/h$	-
r_{ij}	Absolute value of the distance between particle i and j	m
$\mathbf{r}_{ij}$	Vector with the distance between particle i and j	m
r_0	Length scale in the boundary force	m
Δr	Initial particle distance	m
t	Time	s
Δt	Time step used in numerical integration	s
$\mathbf{u}$	Velocity vector	m/s
$\hat{\mathbf{u}}$	Velocity vector adjusted with XSPH	m/s
$\mathbf{u}_{ij}$	Velocity difference vector $\mathbf{u}_i$ - $\mathbf{u}_j$	m/s
u_x, u_y, u_z	Velocity in x, y, or z direction	m/s

$W(\mathbf{x}-\mathbf{x}', h)$	Smoothing function	m^{-2}
W_{ij}	Smoothing function used to calculate interaction between particle i and j	m^{-2}
$\nabla_i W_{ij}$	Spatial derivative of the smoothing function with respect to particle i	m^{-3}
$\mathbf{x}$	Position vector	m
x, y, z	Three directions in Cartesian coordinate system	m

Greek

α	Constant used in viscosity term Π_{ij}	-
γ	Constant used in equation of state (eq. 3.7) $\gamma = 7$	-
ϵ	Constant used in XSPH $0 \leq \epsilon \leq 1$	-
κ	Constant used to determine the influence domain κh, $\kappa = 2$ in this report	-
ν	Kinematic viscosity	m^2/s
Π_{ij}	Influence of viscosity between particle i and j	$\mathrm{m}^5/(\mathrm{kg\ s}^2)$
ρ	Density	kg/m^3
ρ_0	Reference density $\rho_0 = 1000$ kg/m^3	kg/m^3
$\bar{\rho}_{ij}$	Average density $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$	kg/m^3
ω	Wave period	rad/s

Abbreviations

$2D$	2 dimensional
$2Dh$	2 dimensional horizontal
$2Dv$	2 dimensional vertical
$3D$	3 dimensional
SPH	Smoothed Particle Hydrodynamics
XSPH	Variant of SPH which moves a particle not with its own velocity, but with an average between its own velocity and the velocity of particles nearby