Smoothed Particle Hydrodynamics

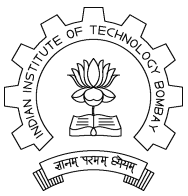
Supervised Learning Programme Report

by

Anand Pratap Singh and Raunak D. Borker
Roll No: 08D01002, 08001001

under the guidance of

Prof. Prabhu Ramachandran



Department of Aerospace Engineering
Indian Institute of Technology, Bombay
Mumbai
Novemeber, 2010

Certificate

Certified that this Supervised Learning Programme Report titled **Smoothed Particle Hydrodynamics** by **Anand Pratap Singh (Roll No. 08D01002) & Raunak D. Borker (Roll no. 08001001)** is approved by me for submission. Certified further that, to the best of my knowledge, the report represents work carried out by the student.

Date: Signature and Name of Guide

Acknowledgement

We would like to thank Kunal Puri for his guidance, support and patience throughout the summer and later. It been fun working with him. We would like to thank our guide Prof. Prabhu Ramachandran for his constant and invaluable support and inspiration. Special mention to Pankaj Pandey for his help with the code and helping us out to find many of the bugs in our code. Finally we would like to thank Prashant, Puneeth, Madhusudan, Amit, Lee and all those who helped us out in developing our python and programming skills.

Anand Pratap Singh
Raunak D. Borker

Abstract

The use of pySPH (a python based SPH solver) to solve a few of the test cases is given in the report. It also briefly highlights the various boundary condition implementation techniques in SPH.

Keywords: *SPH Boundary Conditions, pySPH*

Contents

1.1 Introduction

Smoothed Particle Hydrodynamics is a Meshfree, Lagrangian Particle Method, where each particle is a representative of a discrete part of a problem space. Each particle has associated field variables like pressure, density, position and velocity. On application of conservation of mass, momentum and energy we get the evolution of the physical system.

SPH is used to solve a wide variety of problems. It has application in solving astrophysical problems, free surface flows, shock simulations, fracture of brittle solids and high velocity impact problems concerning the effects of projectiles impacting upon space assets.

A significant aspect of SPH is that any function can be expressed in terms of its values at some particles. SPH involves the kernel estimate of a function. The integral representation of a function in SPH is given by the following equation

$$u(X_i, t) = \int_V W(X_i - X, h)u(X, t)dV$$

And in discrete form can be written as

$$u(X_i, t) = \sum_j \frac{m_j}{\rho_j} W(X_i - X_j, h)u(X_j, t)$$

, where $\triangle V = \frac{m}{\rho}$

This has basically evolved from the property of Translation of the dirac-delta function which is given as follows:

$$\int_{-\infty}^{\infty} f(t)\delta(t - T)\, dt = f(T)$$

Although dirac-delta function doesnt exist in reality, we always have the option of approximating it with some other function. This is exactly what SPH does. The Kernel functions(W) mentioned above, basically are the fucntions which do this. Hence it needs to satisfy the foolowinf propeties:

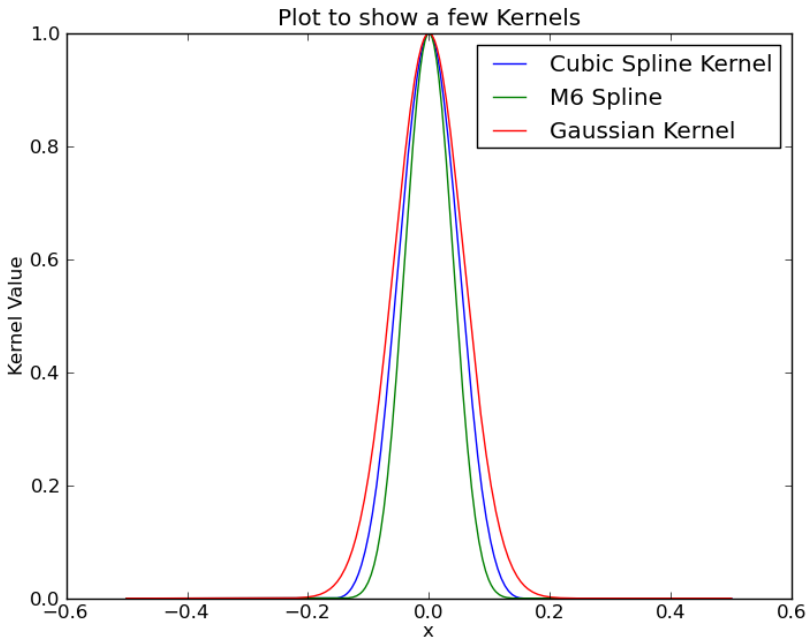
$$\int_v W(x - \mathbf{x}, h)d\mathbf{x} \approx 1$$

$$lim_{h \longrightarrow 0} W(x - \mathbf{x}, h) = \delta(x - \mathbf{x})$$

And a varied number of Kernels have been defined over the years.

Another interpretation to this is that, Kenrel fucntion is an averaging function which finds the value at a particle as an average of its closest neighbours. The average is a weighed one with weights given by the value of the kernel which inversely depends on distance. The averaging smoothens out the properties over the particles. Hence although it is a particle based method it maintains the aspect of continuous domain.

The following is a graph which just compares a few of the regularly used kernel fucntions.



Using these properties we can find the kernel form of gradient of a function. This comes as the following:

$$\frac{du(X_i,t)}{dX} = \sum_j \frac{m_j}{\rho_j} \bigtriangledown W(X_i - X_j, h) u(X_j, t)$$

This equation is valid only in the problem domain and is not applicable at the boundary of the domain. This is because it involves an assumption which is valid only in the domain.

The SPH forms of conservation laws come out to be the following:

Continuity Equation

$$\frac{d\rho(X_i)}{dt} = -\sum_j m_j V(X_j) \bigtriangledown W(X_j - X_i)$$

Momentum Equation

$$\frac{dV(X_i)}{dt} = \sum_j m_j(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2}) \bigtriangledown W(X_j - X_i)$$

Energy Equation without viscosity and heat transfer

$$\frac{dV(X_i)}{dt} = \frac{1}{2}\sum_j m_j(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2})V_{ij} \cdot \bigtriangledown W(X_j - X_i)$$

1.2 Boundary Conditions is SPH

In most engineering problems, the domain of interest is, in general, bounded. The SPH formulations being valid for all interior particles are not necessarily accurate for particles close to the domain boundary since the distribution function (kernel) is truncated by the boundary. Therefore, the application of boundary conditions is problematic in the SPH technique, since SPH approximation no longer produces the second order accuracy. Consequently, the proper and correct boundary treatments have been an ongoing concern for an accurate and successful implementation of the SPH approach in the solution of engineering problems with bounded domains. Improper boundary treatment has two important consequences. The first originates from the penetration of fluid particles into boundary walls, which then leave the bounded domain. The second is that kernel truncation at the boundary will produce errors in the solution.

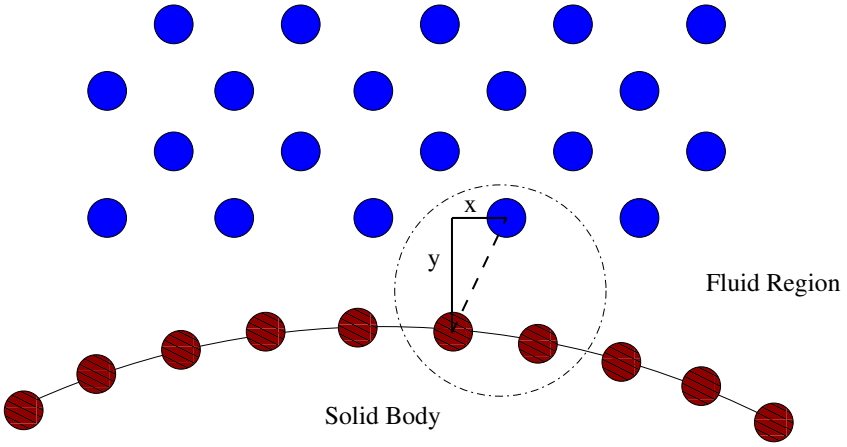
1.2.1 Types of Boundary Conditions

The popularly used SPH Boundary Conditions are briefly described below.

Force Based Method

This technique basically involves creation of boundary particles along the boundaries, and then these particles exert a repulsive force on the fluid particles close to it. The force exerted has a lot of variation depending on the function governing the force. A number of varied functions have been used by different people, and some of these are given as follows:

1. **Monaghan & Kos - Solitary Waves on Cretan Beach**
   As described in Monaghan and Kos (1999) Boundary Particles are assigned a position along the boundary (line of particles). The force due to a boundary particle is calculated based on normal and tangential separation of the fluid particle from it. Hence it is important that we have a technique to calculate the normal and tangential distances between two particles. A visualization of this is shown below:



The force takes the form:

$$f = nR(y)P(x)$$

where
f - force
n - normal to boundary particle
x - tangential distance
y - normal distance

$$R(y) = \frac{A(1-q)}{\sqrt{q}}, q = \frac{y}{2p_o}$$

where, p_o - initial particle spacing
Hence a few particle spacings away form the boundary R(y) tends to 0.
Here

$$A = \frac{1}{h(0.01c^2 + \beta c v_{ab}.n_b)}$$

where $\beta = 1$ if fluid approaching boundary else 0.

$$P(x) = 0.5(1 + cos(\frac{\pi x}{p_o}))$$

for $x < p_o$ else $= 0$.

Only a 2D description is provided by Monaghan & Kos. A 3D figure of this is given on pg.18 in user guide book of SPHysics (http://wiki.manchester.ac.uk/sphysics/index.php/Docum
The no slip condition is implemented by including the boundary particle in calculation of viscous terms in momentum equation.

2. **Monaghan Smoothed Particle Hydrodynamics (Modified)** This techinique is given in Monaghan (2005), it is similar to that given by Monaghan in Monaghan and Kos (1999). The only difference being the way force is computed. The funtioncs $R\&P$ are given new definitions as follows:

$$R(y) = \nabla W(q)\beta,\ \beta = 0.01c^2$$

$$P(x) = 1 - \frac{x}{po}, 0 < x < po$$

$= 0$ otherwise

3. **Monaghan - Free Surface Flows (Lennard-Jones Force)** This i also a force based method but its force computation is quite fiddferent compared to the previous two cases. This is stated is Monaghan (1994).

$$f = D\left(\left(\frac{r_o}{r}\right)^{p_1} - \left(\frac{r_o}{r}\right)^{p_2}\right)\frac{\overrightarrow{r}}{r^2}$$

r distance between fluid particle and boundary particle.
p_1 and p_2 taken as 4 and 2, also $p1 > p2$. p_1 and p_2 taken as 12 & 6 has also worked.
D is dependent on problem can be 5gH, 10gH etc.
r_o initial spacing.
This has a few disadvantages which are stated in the table fpr compartive study of boundary conditions.

If viscous effects need to captured then the boundary particles are included in the viscous stress calculation in all of these force based methods. Initializing routine should ensure no particle is too close to boundary near curved surface else the force is too high. Initial equilibrium state is assured by a damping term in momentum equation $-\tau v$.

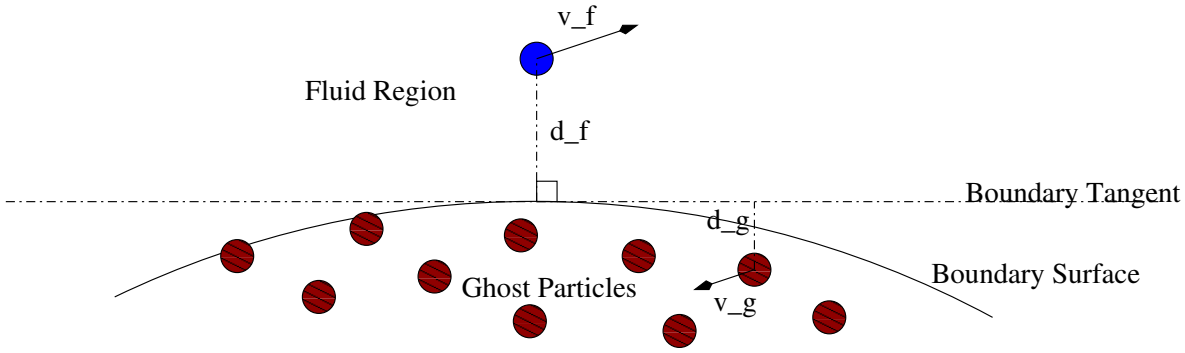
Ghost Particles Method

When a particle is close to a boundary (distance smaller than smoothing length) then a virtual (ghost) particle is created outside the boundary by taking a image of the given partcle about the boundary. Both the particles have the same properties except velocities which are in the opposite direction. This helps prevent penetration as it takes care of the

normal valocity very well. Another way (Single boundary tangent technique) in which this is implemented is that ghost particles are created during the particle generation itself. And based on the normal distance from the tangent to the boundary corresponding to this particle their velocity is calculated. The formula is given as follows:

$$\overrightarrow{v_g} = -\frac{d_g}{d_f}\overrightarrow{v_f}$$

This method has been succesfully used by a number of researchers. The only problem with it is that it is a bit complicated to implement as it involves generation of particles based on distance and angle. Hence it is required to calculate tangent and normal to particles. A visualization of the Single boundary tangent treatment is shown below:



A variant of this is the **Multiple Boundary Tangent Method** described in M. Yildiz and Suleman (2008) & SPHysics. This basically involves the following steps:

1. At each time step, for all boundary particles, tangent lines are computed.
2. Given that each boundary particle has fluid particles in its influence domain as neighbors, these fluid particles are mirrored with respect to the tangent line of the corresponding boundary particle.
3. During the SPH summation over ghost particles for a fluid particle with a boundary truncation, the mass of the ghost particles are divided by the number of corresponding over-creations.

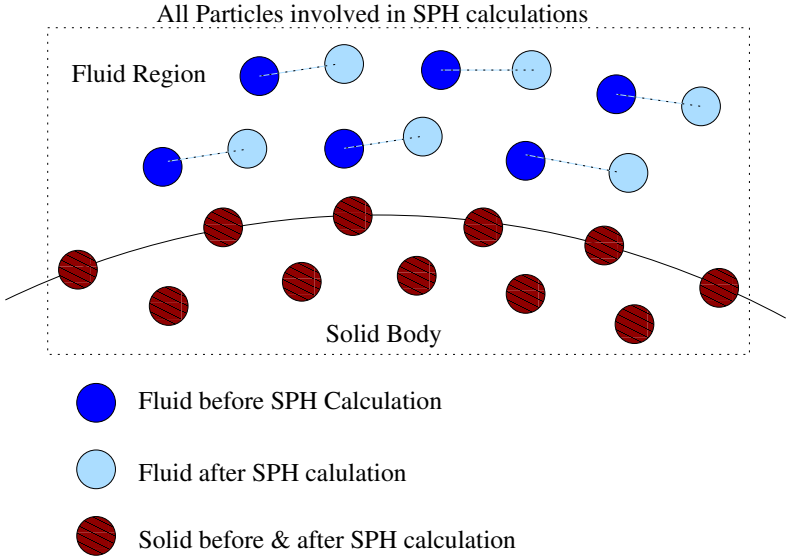
A Combination of Boundary Force & Ghost Particles

This is described in Ferarri *et al.* (2009). He uses boundary particles as in boundary force methods but no force is applied. Instead we compute a flux between fluid particle and each of the boundary particles via the standard kernel summation by locally mirroring particle at each boundary particle via point symmetry. The contribution of the

wall therefore depends on the position and velocity of particle, but not on the ones of the neighbors. In the Monaghan Boundary Force technique the time step is limited due to large boundary force which may lead to instability in the solution. In the present technique covers up for this to a large extent. Another advantage of our new approach for the boundary conditions is that no parameters must be tuned.

Dynamic Boundary Particles Technique

This technique was first used by Dalrymple & Knio in Dalrymple and Knio (2000). It is also well describe din A. J. C. Crespo and Dalrymple (2007). Dynamic Boundary Particles method basically is computationally simpler and less expensive than the other methods. Here the boundary particles are included in all the SPH calculations but their position is fixed (i.e. it is externally position is fixed and velocity kept at 0.). Hence it is just like any other calculation on the given particles which are fixed. The advantage is that we dont require creating new ghost particles neither is the evaluation of distance required as in repulsive force boundaries. A graphical visualization this is shown below:

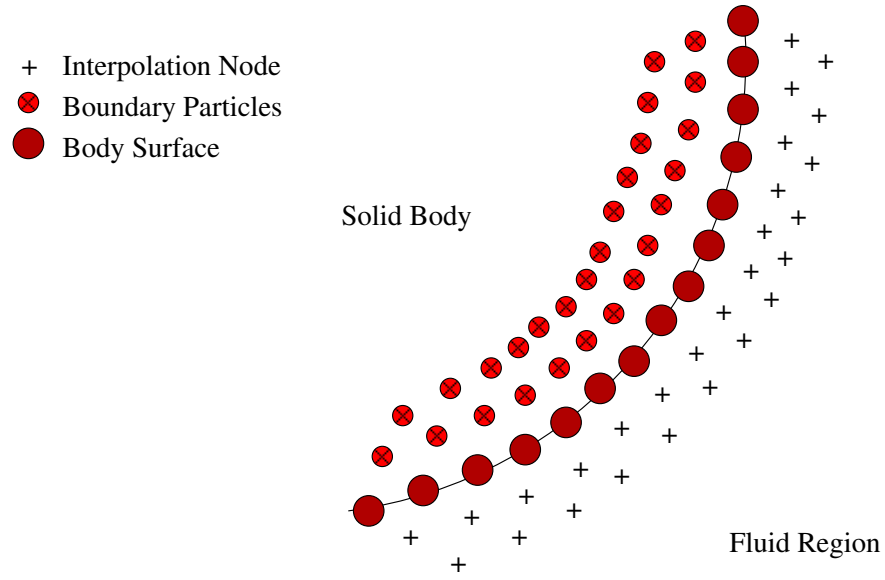


One of the problems that could arise in DBC is that of fluid particle penetration into boundary (if the particles approach too fast). Also it is too sticky fluid particles do not drain completely from vertical walls.

Enhanced Boundary Treatment

This is a very recent technique and is described in Colicchio and Graziani (2009). Here they approximate body profile by body nodes using spline. Then inteprolation body nodes

are placed of the fluid side of boundary and are then mirrored out into the body profile about the body curve. This has the advantage that it does not involve dynamic particle creation as in ghost particles. They have succesfully solved the Dam-Break against a sharp object problem. This problem is a good indicator as it involves sharp geometry as well as high reynolds number flow. A visualization of this is given in the following figure:



A table comparing these techniques is given on the following page.

Table 1.1: Comparison of Boundary Condition Implementation Techniques

METHOD	BRIEF	ADVANTAGE	DISADVANTAGE	PROBLEM$
Lennard-Jones Force	Repulsive Force exerting Particles with force of Lennard-Jones form.	Prevents Penetration	Causes Oscillations in force distribution when particle moving parallel to boundary. Doesnt eliminate Kernel truncation. Distortion in flow close to solid boundary	
Boundary Particles exerting specific Force	Repulsive Force exerting particles	Prevents Penetration. Tries to prevent Ripples due to force(does to some extent). Easier to implement for complicated geometry	For simple geomtries ghost particle gives better results. Produces Bouncy effect. Doesnt eliminate Kernel truncation. Distortion in flow close to solid boundary	Solitary Wave
Ghost Particles	Dynamic generation of temporary particles at every time step	Prevents Penetration. beneficial for simple geometry	Difficult for complicated Geometry,as there could be multiple normal vectors possible to a bondary surface. Doesnt eliminate Kernel truncation. Distortion in flow close to solid boundary.	Most of The problems can be solved
Multiple Boundary Tangent	Similar to ghost, but multiple reflections which are later averaged out.	Treats Curved Boundary Surfaces	Complicated to implement for a general case	Lid-driven Cavity, Flow past cylinder, claim free surface possible
Ghost Particles + Boundary Force	Is a combo of ghost and boundary force techniques.	Contribution depends on the position and velocity of particle, but not on the ones of the neighbors. Accurate, little oscillatory pressure elds in proximity of uid-solid interfaces, can any complex geometry	Increases number of computations	3D Dam Break
Dynamic Boundary Particles	Normal Boundary Particles included in SPH	Computational very cheap	Too stickyfluid particles do not drain completely from vertical walls. May not prevent penetration for fluid approaching with high velocity.	Wave Maker, Dam Break
Enhanced Boundary method	Normal Boundary Particles included in SPH	Computational very cheap	Possible only in 2D	Dam-Break against a sharp object

1.3 Sod's Shock Tube Problem

Here we have solved the Sod's Shock Tube Problem using our own code. The initial conditions are as follows:

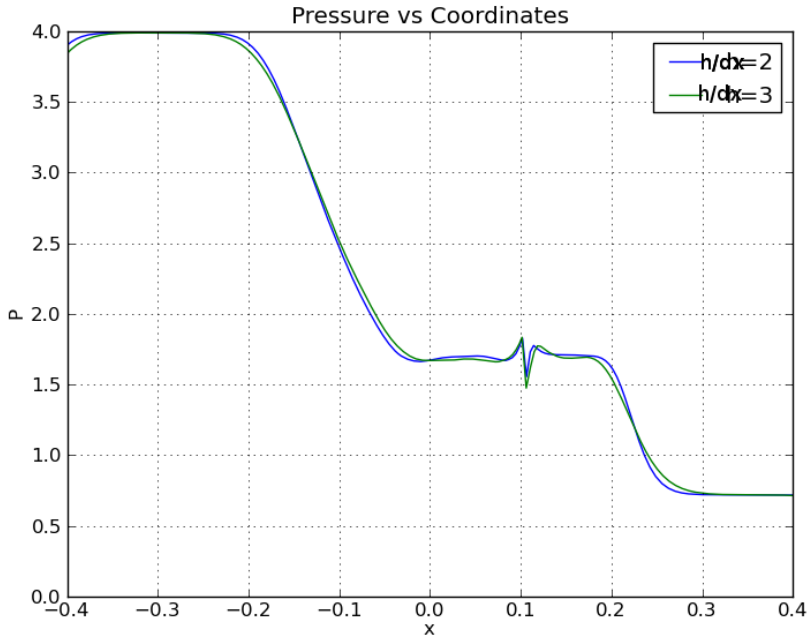
-0.6 < x < 0
3200 Particles
$\rho = 0.25kg/m^3$
Thermal energy(e) $= 2.5J$

for **0 < x < 0.6**
800 Particles
$\rho = 1.0kg/m^3$
Thermal energy(e) $= 1.795J$

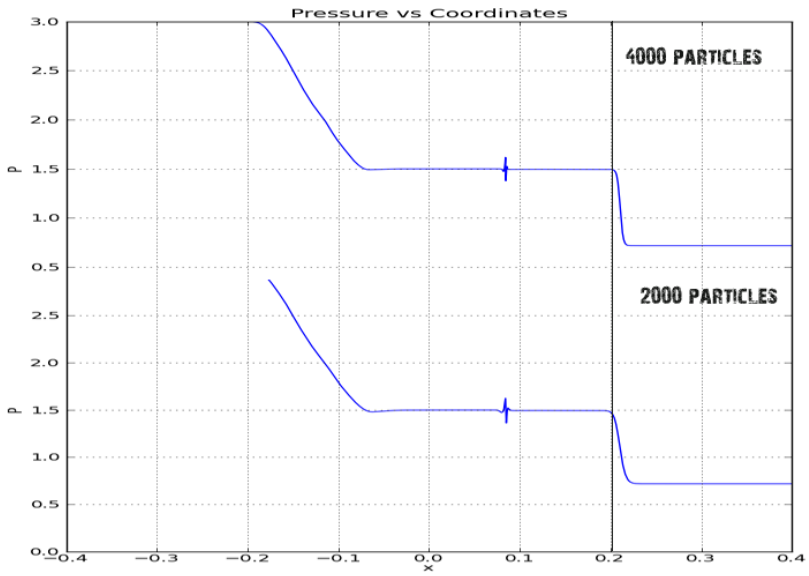
$pressure = (\gamma - 1) \times \rho \times e$

$$\frac{h}{dx} = 2$$

This initial state can be produced by having a diaphragm in the middle of the tube. The particles are at rest initially. At t=0 the diaphragm is broken.
This was also performed with 1600:400 particles as well. h/dx was set to 2 and 3. The results are provided as follows.



It is observed that the result with a smaller h/dx ratio gives results closer to analytical ones. Genrally to obtain better resolution we use a smaller h and higher number of particles. They are correalted and hence a limit on number of particles limits h. Hence in our case for same particles better results were obtained for a lower h. A good description of smoothing length and it's selection is give in B. Weyssow (2008) .



The following graphs compare the two cases with different number of particles. It is clearly observed that more particles give sharper solution and hence preferred. But the disadvantage is that it increases the number of computations.

1.4 Sample code in pySPH

pySPH is an Open-Source Smoothed Particle Hydrodynamics Code developed in python. This can be obtained from pySPH. This code incorporates the following features:

- Choice of time integration schemes: Euler, Runge-Kutta, predictor-corrector
- Choice of Kernel functions:Gaussian, Harmonic, Cubic Spline
- Kernel gradient correction
- Choice of two types of solid boundary condition: dynamic and repulsive force
- Choice of viscosity treatments:Artificial Viscosity, Laminar Viscosity
- Visualization routines using Matlab or ParaView
- Capability for different Equations of State
- Choice of frequency of data storage
- OPEN SOURCE: Developed in Python

Basic Imports

```
import numpy
import pysph.base.api as base
import pysph.solver.api as solver
import pysph.sph.api as sph

Fluid = base.ParticleType.Fluid
Solid = base.ParticleType.Solid
```

Defining Parameters

```
h = 0.0260
dx = dy = h/1.3
ro = 1000.0
co = 65.0
gamma = 7.0
muu = 1.0e-6
alpha = 0.1
eps = 0.5
```

```
fluid_collumn_height = 1.0
fluid_collumn_width  = 0.5
container_height = 1.5
container_width  = 4.0

B = co*co*ro/gamma
```

Setting Up Geometry

In our case we created a staggered grid, which can be created using basic python utility.

In a 2D case following parameters need to be defined:

```
#Smoothing length
hb = numpy.ones_like(xb)*h
#mass
mb = numpy.ones_like(xb)*dx/2.*dy/2.*ro
#density
rhob = numpy.ones_like(xb) * ro
#viscosity coeff
mu = numpy.ones_like(xb)*muu
#Speed of sound
cb = numpy.ones_like(xb)*co
```

Then the particle arrays can be defined as

```
#a new type can be added in particles_type.py
#and it should also be modified in particles.py
boundary = base.get_particle_array(name=name, type=Solid,
                                        x=xb, y=yb, h=hb, rho=rhob, cs=cb,
                                        m=mb, mu=mu)
```

Similary fluids particle arrays can be created.

Setting the application

```
# Define a application
app = solver.Application()
#takes input from command line and process it, for all
#the available set of inputs are
```

```
app.process_command_line()
#Generate the particles, argument is a function
#which give a list of particle arrays
particles = app.create_particles(get_particles)
#set the kernel
s = solver.Solver(base.HarmonicKernel(dim=2, n=3),
                  solver.RK2Integrator)
```

Now we need to add the operation, they are the mathematical operation that we need to perform between the particles. This includes all the equations needed to solve the physical system. They are defined as follows.

```
#Equation of state
s.add_operation(solver.SPHAssignment(

        sph.TaitEquation(co=co, ro=ro),
        on_types=[Fluid, Solid],
        updates=['p', 'cs'],
        id='eos')

                )

#Continuity equation
s.add_operation(solver.SPHSummationODE(

        sph.SPHDensityRate(),
        on_types=[Fluid, Solid], from_types=[Fluid, Solid],
        updates=['rho'], id='density')

                )

#momentum equation
s.add_operation(solver.SPHSummationODE(

        sph.MomentumEquation(alpha=alpha, beta=0.0),
        on_types=[Fluid], from_types=[Fluid, Solid],
        updates=['u','v'], id='mom')

                )
```

```
#Gravity force
s.add_operation(solver.SPHSimpleODE(

        sph.GravityForce(gy=-9.81),
        on_types=[Fluid],
        updates=['u','v'],id='gravity')

                )

#XSPH correction
s.add_operation(solver.SPHSummationODE(

        sph.XSPHCorrection(eps=eps),
        on_types=[Fluid], from_types=[Fluid],
        updates=['x','y'], id='xsph')

                )

#Position stepping
s.add_operation(solver.SPHSimpleODE(

        sph.PositionStepping(),
        on_types=[Fluid],
        updates=['x','y'], id='step')
```

If you look carefully you can understand the importance of from types and on types, which can mess up rsults if go are wrong. For example, position stepping is need to be done only for types fluid, but continuity equation is solved for both solid and fluid and are affected by both of them etc.

Setting final run parameters

```
s.set_final_time(10.0)
s.set_time_step(1e-4)
app.set_solver(s)
#after how many steps you need to dump the output
s.set_print_freq(10)
```

Finally you can run using app.run(). Have fun with pySPH.

1.5 Simulation of Test Cases using pySPH

1.5.1 2D Twin Dam Break in a Box

This problem involves the simulation of the collapse of two vertical columns of a fluid (called dam) under the influence of gravity in a container.

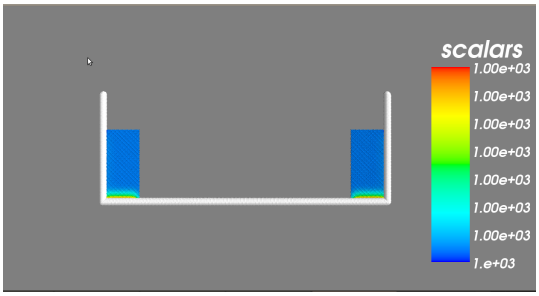
Geometrical and Simulation Parameters

Container: Width=1.5m, Height=4m
Dams: Width=0.5m, Height=1m
Smoothing length h = 0.0390
h/dx = 1.3 as used by State of the art SPH gives dx = dy =0.03m
Kernel Harmonic Kernel
Boundary Conditions Dynamic Boundary Conditions
Artificial Viscosity using Monaghan Artificial Viscosity $\alpha = 0.25$
XSPH correction epsilon 0.5
Total Particles
Time Step 1e-4 Total Time 10s

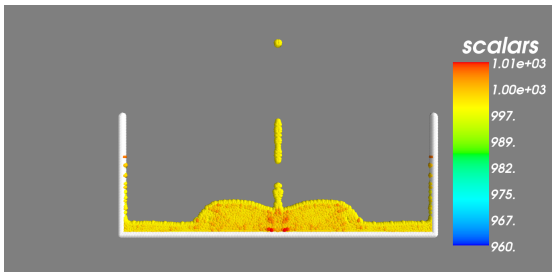
Results

Table 1.2: 2D Twin Dam Break Density Plot, $\alpha = 0.250h = 0.0260$

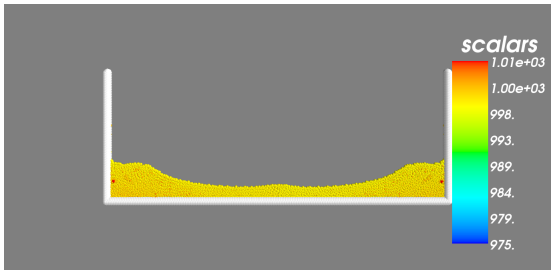
t = 0.0s



t = 1.5s



t = 2.5s



t = 3.5s

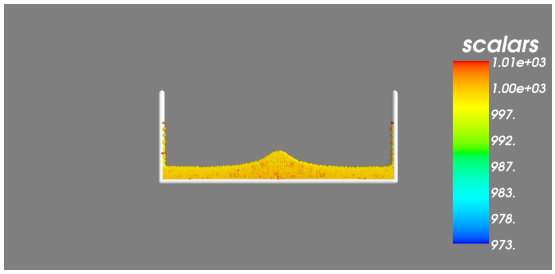
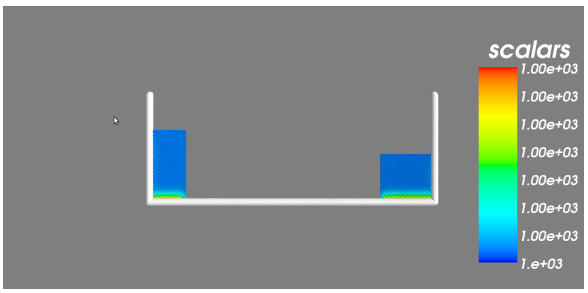
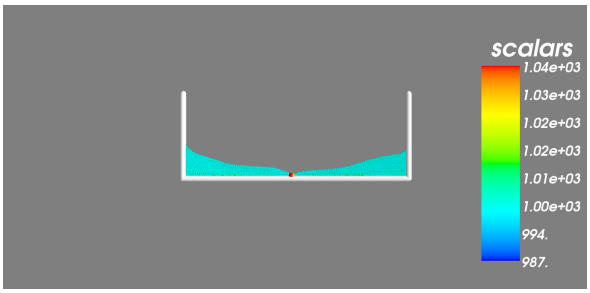


Table 1.3: 2D Two Non-Identical Dam Break Density Plot, $\alpha = 0.25h = 0.026$

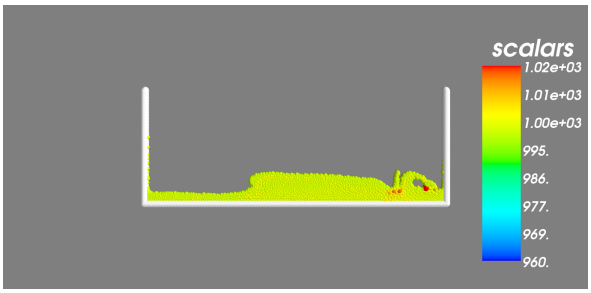
t = 0.0s



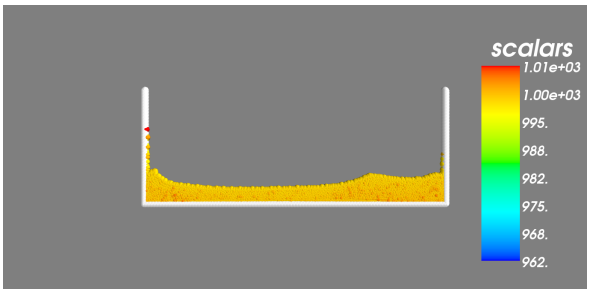
t = 0.62s



t = 1.0s



t = 1.475s



The figures given above represent two cases. The first one is the one described above. The second figure shows a lsight modification where the dams are non-identical.

Conclusion

We observe that in both of the above cases the density approximately remains constant. It is a bit higher only in regions where there is a bit of clogging or where the particles collide. The density remaining approximately constant is an indication of correct solution as the assumption we have made is that velocity of sound is high and hence the flow can be treated incompressible.

1.5.2 2D Fall of fluid column into another

This problem basically involves the simulation of the fall of a fluid column under the effect of gravity into a container containing the same fluid. The particles were placed ina staggered grid manner.

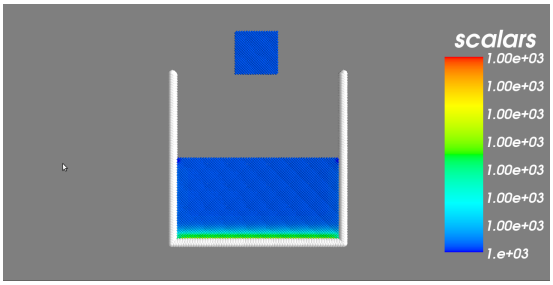
Geometrical and Simulation Parameters

Container: Width=2m, Height=2m
Fluid in Conatiner: Width=2m, Height=1m
Falling Fluid: Width=0.5m Height=0.5m
Falling Height: 2m
Smoothing length h = 0.0390
h/dx = 1.3 as used by State of the art SPH gives dx = dy =0.03m
Kernel Harmonic Kernel
Boundary Conditions Dynamic Boundary Conditions
Artificial Viscosity using Monaghan Artificial Viscosity $\alpha = 0.5, 0.25$
XSPH correction epsilon 0.5
Total Particles 4389 fluid, 578 fall fluid, 403 boundary
Time Step 1e-4 Total Time 10s

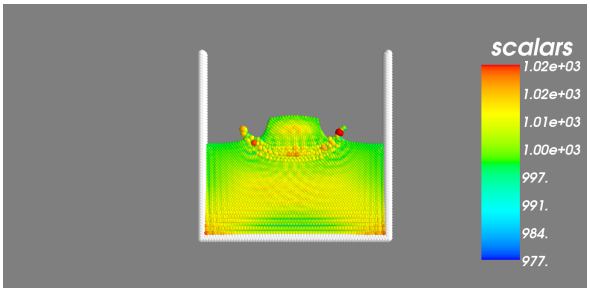
Results

Table 1.4: Density Plot for $\alpha = 0.1$

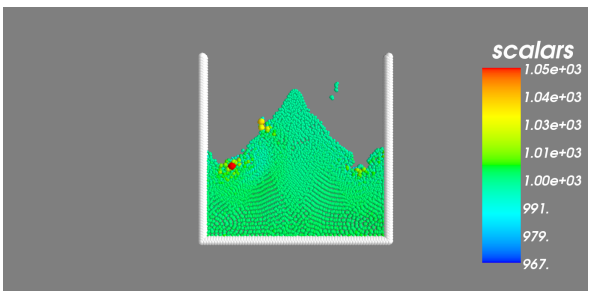
t = 0.0s



t = 0.5s



t = 1.5s



t = 2.5s

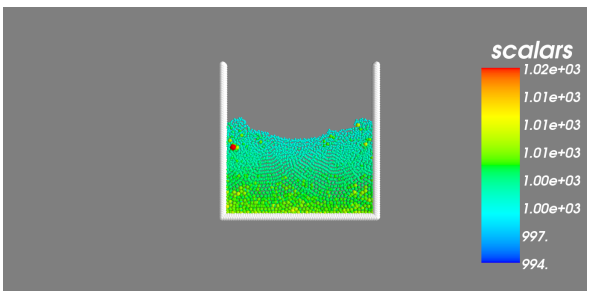
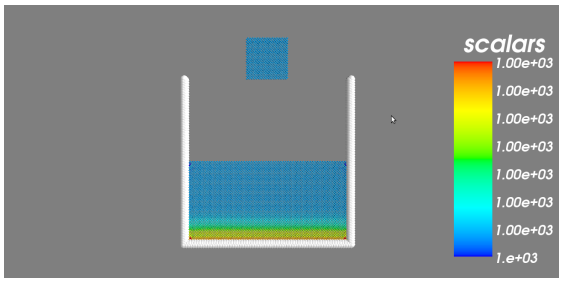
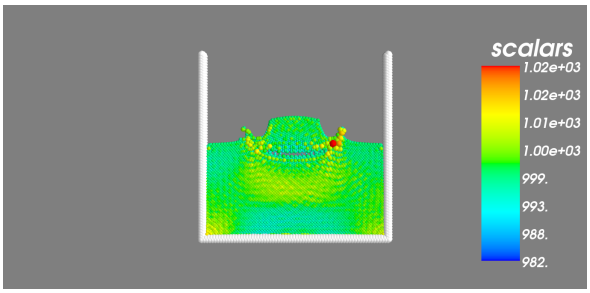


Table 1.5: Density Plot for $\alpha = 0.05$

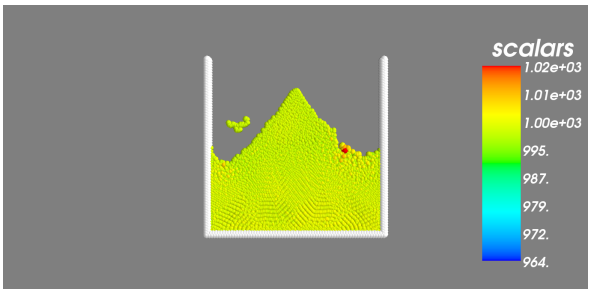
t = 0.0s



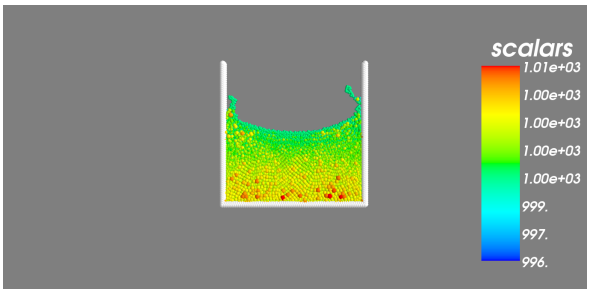
t = 0.5s



t = 1.5s



t = 2.5s



Conclusion

Both the cases seem very similar. The only difference as expected is that in the case of lower artificial viscosity the particles stickiness is lower and hence they split more. Even the particles sticking to the wall is lower in the case of lower viscosity.

1.5.3 2D Dam Break with an Obstacle

This problem involves the simulation of the collapse of two vertical columns of a fluid (called dam) under the influence of gravity in a container.

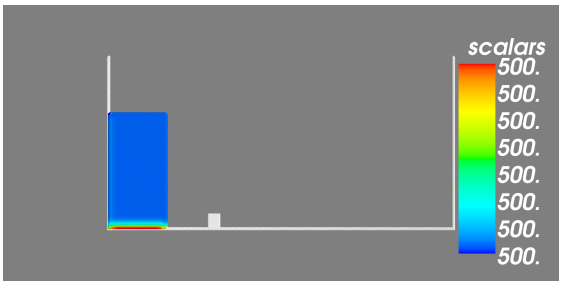
Geometrical and Simulation Parameters

Container: Width=6.0m, Height=3.0m
Left Fluid Particles: Width=2.0m, Height=1.0m
Obstacle Solid Particles: Width=6dx m, Height=8dx m
Smoothing length h = 0.0390
h/dx = 1.3 as used by State of the art SPH gives dx = dy = 0.029m
Kernel Harmonic Kernel
Boundary Conditions Dynamic Boundary Conditions
Artificial Viscosity using Monaghan Artificial Viscosity $\alpha = 0.4$
XSPH correction epsilon 0.5
Total Particles
Time Step 1e-4 Total Time 8.2s

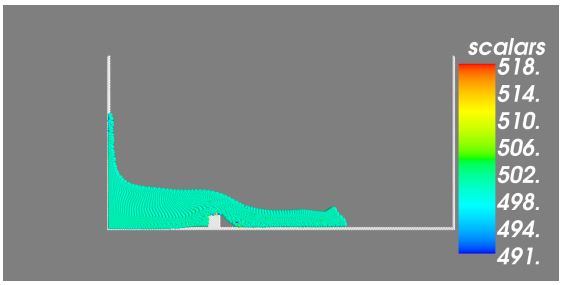
Results

Table 1.6: Density Plot for $\alpha = 0.4$

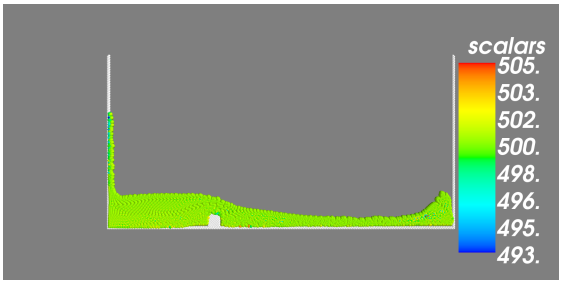
t = 0.0s



t = 1.0s



t = 2.0s



t = 8.27s

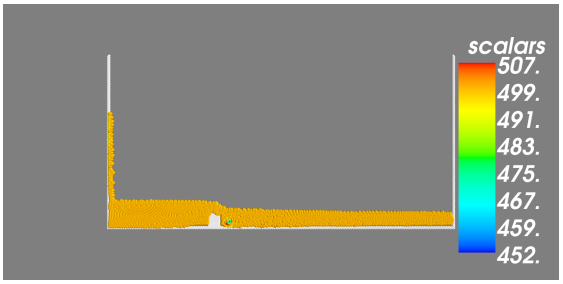
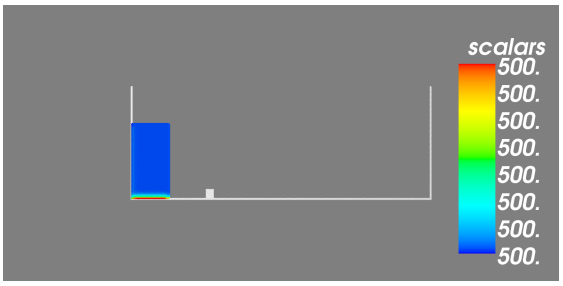


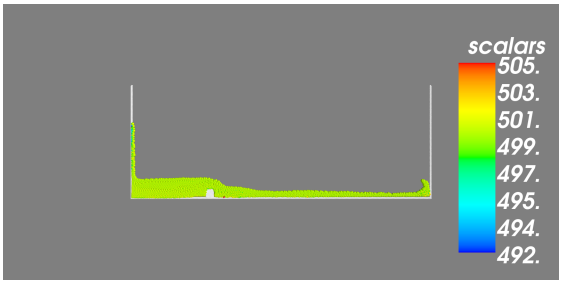
Table 1.7: Density Plot for $\alpha = 0.3$

t = 0.0s

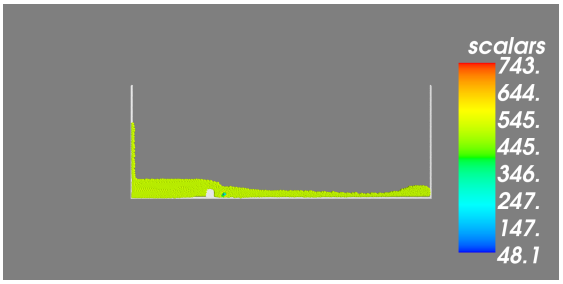


t = 1.0s

t = 2.0s



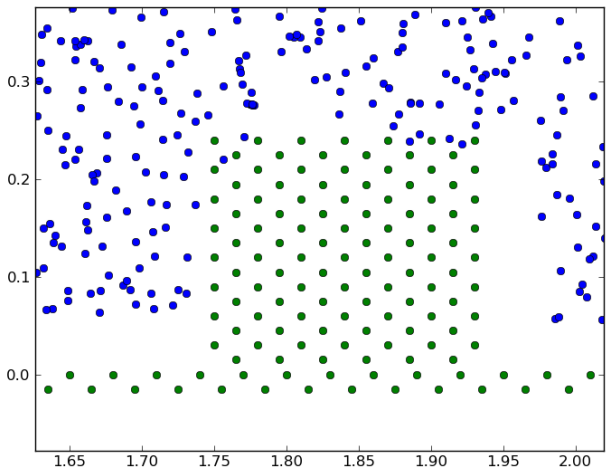
t = 2.8s



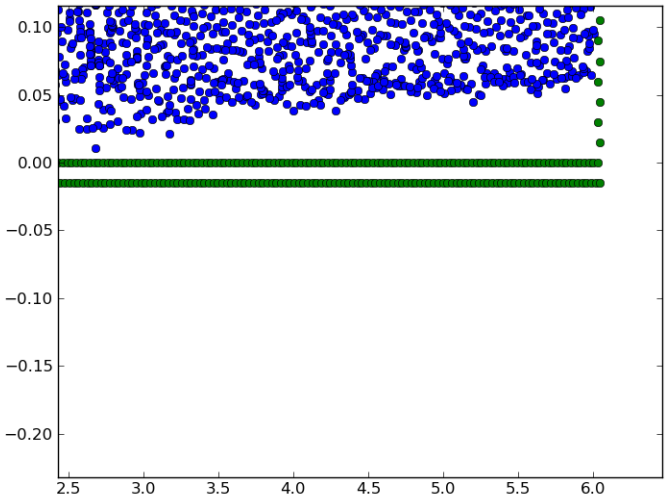
Conclusion

We can conclude from the results that, density remains almost constant throughout the domain, which implies that artificial compressibility is working fine. There is a little amount of penetration taking place, in the obtacle(see figure below). At the bottom of the dam, fluid particles are slight shifted upwards (this is of the order of $h/2$) and shift increases as we move in the right direction(see figure below), a suspected reason of this might be due to the component of boundary force acting in the upward direction, but can be said for sure. Also with decrese in viscosity coefficient fluid particles doesnt seperate and remain intact.

Penetration of fluid particles into the obstacle, $\alpha = 0.4$



Fluid particles are shifted upwards, $\alpha = 0.4$



1.5.4 2D Dam Break over a Wet Bed

This problem involves the simulation of the collapse of two vertical columns of a fluid (called dam) under the influence of gravity in a container.

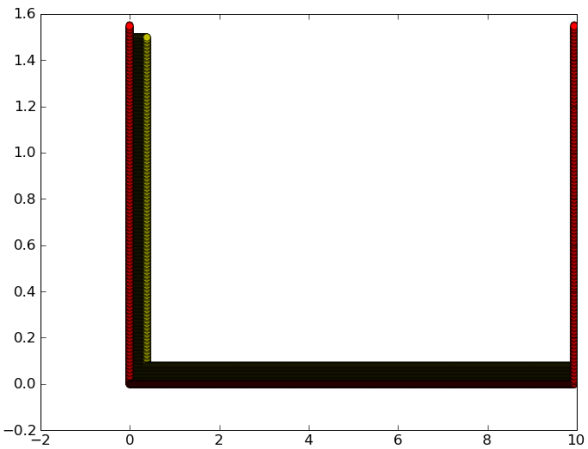
Geometrical and Simulation Parameters

The case is described in M. Gmez-Gesteira1 and Crespo (2010)

Container: Width=9.93m, Height=1.55m
Left Fluid Particles: Width=0.38m, Height=1.5m
Right Fluid Particles: Width=9.558m, Height=0.078m
Gate Partciles: dx=dx/4, Velocity = 1.5m/s
Smoothing length h = 0.006
h/dx = 1.2 as used by State of the art SPH gives dx = dy =0.005m
Kernel Harmonic Kernel
Boundary Conditions Dynamic Boundary Conditions
Artificial Viscosity using Monaghan Artificial Viscosity $\alpha = 0.03$
XSPH correction epsilon 0.5
Total Particles
Time Step 1e-4 Total Time 10s

Initial Condition

Table 1.8: 2D Dam Break over a wet bed $\alpha = 0.03$
t = 0.0s



Note

This case run just for 10 steps before giving a JTolerance error. The reason for error is not known. It was suspected thas it might be due to the high velocity of gate particles but even after decreasing it there was no change. Changing cell size and JTolerance value also didnt help.

1.6 Simulation of Flow Past a Bluff Body using pySPH

This problems involves the flow past a Bluff Body of a fluid with some initial velocity in a channel. The problem basically is a like a poiseuille's flow with a bluff body in its path. We used Dynamic Boundary Particles for this problem.

1.6.1 Geometrical and Simulation Parameters

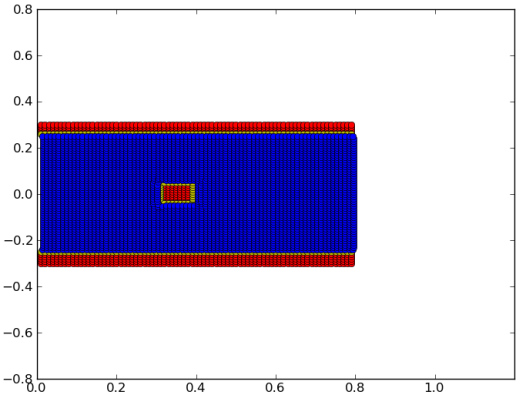
Length of Tank = 1.4m
Width of Tank = 0.5m
Thickness of wall = 0.05m
Bluff Body = 0.1m × 0.1m Distance at which Bluff Body is placed = 0.3m from the start of the channel
dx = dy = 0.01m
Total Number of Particles = 959 boundary, 3860 fluid

Kernel = Cubic Spline Kernel
Time Step = 0.0001sec for 0.5s Boundary Conditions = Dynamic Particles Boundary Conditions Viscosity = Inviscid Flow

1.6.2 Results

Table 1.9:

t = 0.005s



t = 0.075s

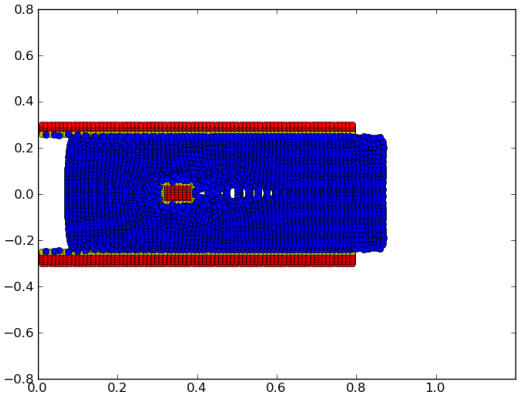
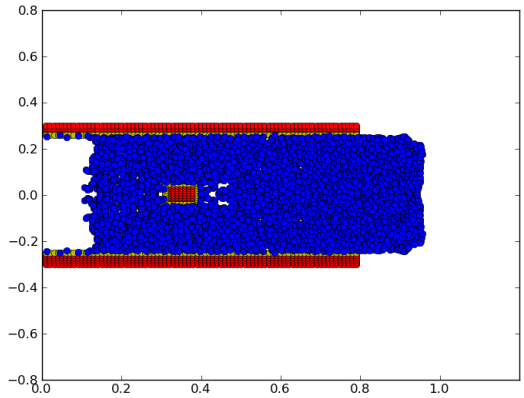
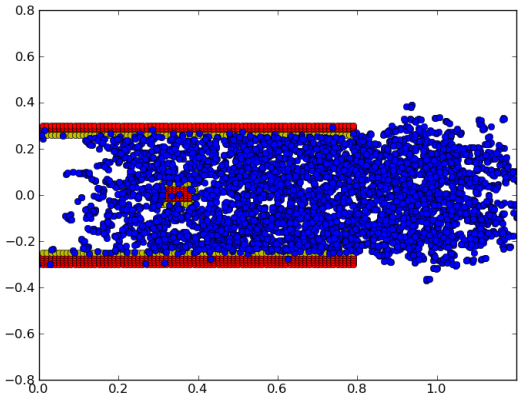


Table 1.10:

t = 0.15s



t = .3s



1.6.3 Conclusion

The implementation of Periodic Boundary Conditions is required for this problem as it is an infinite channel. We tried to implement such a condition by eliminating the particles which have reached the end after the length where flow has developed, and generating new particles with te inlet properties at the inlet. But running this we got an error of Jtolerance.

On running it without the periodic boundary conditions decent outputs are observed for a few time steps before the particles go haywire. An important observation after this happens is that of the shortcoming of the Dynamic Boundary Particle Conditions. It is seen in the following graph that as the particles move haywire and have large velocities many of them have just penetrated the solid boundary. Hence DBP's are not capable of preventing penetration as mentioned in the previous section on boundary conditions.

Bibliography

1. **A. J. C. Crespo, M. G.-G.** and **R. A. Dalrymple** (2007). Boundary conditions generated by dynamic particles in sph methods. *CMC*, **5**, 173–184.

2. **B. Weyssow, Q. V., C. Toniolo** (2008). On determining the smoothing length in the smoothed particle hydrodynamics (sph) description of fluids. *35th EPS Conference on Plasma Phys. Hersonissos*, **32D**.

3. **Colicchio, S. M. M. A. A. C. G.** and **G. Graziani** (2009). Enhanced boundary treatment in 2d smoothed particle hydrodynamics models. *Congress of the Italian Association for Theoretical and Applied Mechanics*, **19**.

4. **Dalrymple, R. A.** and **O. Knio** (2000). Sph modelling of water waves. *Proc. Coastal Dynamics*, **2**, 1082–1091.

5. **Ferarri, A.**, **M. Dumbser**, **E. F. Toro**, and **A. Armanini** (2009). A new 3d parallel sph scheme for free surface flows. *Computers and Fluids*, **38**, 12031217.

6. **M. Gmez-Gesteira1, R. A. D.** and **A. J. C. Crespo** (2010). State-of-the-art of classical sph for free-surface flows. *Journal of Hydraulic Research*, **48**, 627.

7. **M. Yildiz, R. A. R.** and **A. Suleman** (2008). Sph with the multiple boundary tangent method. *International Journal Numerical Method Engineering*.

8. **Monaghan, J.** (1994). Simulating free surface flows with sph. *Journal of Computational Physics*, **110**, 399–406.

9. **Monaghan, J.** (2005). Smoothed particle hydrodynamics. *Institue of Physics Publishing, Rep. Prog. Phys.*, **68**, 17031759.

10. **Monaghan, J.** and **A. Kos** (1999). Solitary waves on a cretan beach. *Journal of Waterway Port, Coastal, and Ocean Engineering*, **125**, 145–155.

11. **pySPH** (). *pySPH documentation*. Prabhu R., Kunal, Pankaj, http://code.google.com/p/pysph/.

12. **SPHysics** (). *SPHysics Guide Book, Second International Workshop*, 2.200 edition.