Smoothed Particle Hydrodynamics - Lab course

Final Report

Dwaipayan Roy Chowdhury

5360085

dwaipayan0502@gmail.com

February 10, 2023

Contents

1

Introduction

Fluid simulation is a computer graphics technique used to realistically depict the behavior of liquids and gases in a virtual environment. It involves solving mathematical models that describe the motion, pressure, and density of fluids under various conditions. The simulation results can be used in various fields such as entertainment, engineering, and scientific research to visualize and analyze fluid behavior in real-life scenarios. The simulations can also incorporate various physical phenomena such as viscosity, turbulence, and surface tension to achieve more realistic results.

Smoothed Particle Hydrodynamics (SPH) is a numerical simulation technique used in astrophysics, engineering and computer graphics. It is a Lagrangian method for solving the equations of fluid dynamics, in which the fluid is represented as a collection of individual particles, or smoothed masses. The fluid properties, such as density, velocity and pressure, are estimated by averaging the properties of the surrounding particles. This method is especially useful for modeling complex fluid flow problems with free surface flows, like ocean waves and the flow of liquids in containers. This discretization allows the fluid behavior to be represented as a numerical solution using mathematical models, making it possible to simulate fluid motion and properties using computer algorithms.

In this report, we present a brief development of SPH discretization for fluid solver development. We develop a simple solver of some wet fluid nodes within a boundary (dry nodes). In Lagrangian technique we consider the nodes as fluid particles that have density, mass, volume etc. We also talk about the algorithm used to run the simulation and test strategies used. We perform some simulations to discuss about the influence of stiffness constant (K), viscosity and discretization timestep on average density. Finally, we conclude by presenting some simulation plots over time.

2

Concepts

2.1 SPH Discretization

A particle based simulation is a mesh-free method that is purely based on lagrangian description that flows depending on some dynamic properties like position $x \in R^3$, velocity $v = \frac{dx}{dt} \in R^3$, acceleration $a = \frac{d^2x}{dt^2} \in R^3$ and mass $m \in R$. Since it is not parametric, the fluid properties are often realized as a summation of some weights generated using a kernel/weighing function. This is a typical concept of non-parametric value decomposition. Now let us talk briefly about the kernel function and some if its interesting properties.

Usually SPH is more understood as a discretization technique rather than a simulation method. A dirac (δ) gives sudden impulse to specific points and the generalized definition of dirac can be written as [1],

$$\delta(r) = \begin{cases} 0 & \text{if } r \neq 0 \\ \infty & \text{if } r = 0 \end{cases}$$

and thus satisfies $\int \delta(r)dv = 1$. This convolution supports the compactness of the kernel and this is a usual notion in smoothing function that the cumulative summation of its values should result in unity. Let's consider the function as $A(x)$ which is identical to the δ distribution, thus can be written as [2]

$$A(x) = (A * \delta)(x) = \int A(x')\delta(x - x')dv'. \tag{2.1}$$

Here, dv' denoted the discrete volume of the particle represented using x'. Now, our task is to discretize the δ distribution which is a gaussian distribution with zero variance. So, we approximate this to a kernel function $W : R^d \times R^+ \implies R$. And the approximation function can be written as,

$$A(x) \approx (A * W)(x) = \int A(x')W(x - x', h)dv' \tag{2.2}$$

In this discretization, we added one more parameter h referring to smoothing length. It holds two important conditions i.e. normalization condition and dirac-δ condition. Now, to attach this mathematical formulation to the properties like density, energy and momentum, we need an interpretable formula in terms of the properties. We, thus write the kernel as (we use cubic spline with compact support between $2h - 5h$ [1][2],

$$W(q) = \alpha \begin{cases} (2-q)^3 - 4(1-q)^3 & \text{if } 0 \leq q < 1 \\ (2-q)^3 & \text{if } 1 \leq q < 2 \textbf{ where, } q = \frac{\|x_j - x_i\|}{h} \\ 0 & \text{if } q \geq 2 \end{cases} \tag{2.3}$$

where, $q = \frac{\|x_j - x_i\|}{h}$. Since this is a 2D simulation, we use alpha value as $\alpha = \frac{5}{14\pi h^2}$. We used kernel support as $2h$ and particle spacing as h. We also need the first derivative value in properties which denote rate of change with time. So, we have the first derivative of the kernel defined as [1],

$$\nabla W(q) = \frac{x_j - x_i}{6h^2\|x_j - x_i\|} \begin{cases} -3(2-q)^2 + 12(1-q)^2 & \text{if } 0 \leq q < 1 \\ -3(2-q)^2 & \text{if } 1 \leq q < 2 \ 0 \\ \text{if } q \geq 2 \end{cases} \tag{2.4}$$

Here, $q = \frac{\|x_j - x_i\|}{h}$. This will lead to to approximation results that can represent properties of fluid. Let us now use the reformed weighing kernel to the properties formulation. This is where SPH discretization helps us build the discrete property features of a continuous function. [1]

$$\nabla A_i = \Sigma_j \frac{m_j}{\rho_j} A_j \nabla W_{ij} \tag{2.5}$$

$$\nabla^2 A_i = \Sigma_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij} \tag{2.6}$$

$$\nabla A_i = \rho_i \Sigma j m_j (\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}) \nabla W_{ij} \tag{2.7}$$

$$\nabla^2 A_i = 2\rho_i \Sigma_j \frac{m_j}{\rho_j} \frac{A_i \cdot x_{ij}}{x_{ij} \cdot x_{ij} + 0.01h^2} \nabla W_{ij} \tag{2.8}$$

Now, we include the derivatives into our property functions. We use explicit form for density calculation which is comparatively accurate and can be error-prone when the neighborhood is incomplete. The density can be written as,

$$\rho_i = \Sigma_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \Sigma_j m_j W_{ij} \tag{2.9}$$

This equation sums over all neighboring particles within a smoothing length h of particle i to compute the local density. The choice of smoothing kernel and smoothing length affects the accuracy and stability of the simulation.[2]

Now that we know about the density, we can calculate the pressure with this simple formula given as,

$$p_i = k(\frac{\rho_i}{\rho_0} - 1) \tag{2.10}$$

Following the earlier derivatives, we can write the accelerations easily. The following algorithm 2 presents the complete flowchart for the solution of the SPH simulation.

2.2 Neighborhood search

In SPH simulation, one of the open research work is happening in the field of neighborhood algorithm to build the neighborlist. We intend to find all i-j pairs with $r_{ij} < r_c$. A spatial grid neighborhood algorithm is a method for efficiently finding neighboring particles in a particle-based simulation, such as molecular dynamics or smoothed particle hydrodynamics (SPH). The basic idea is to divide the simulation space into a grid of cells and assign each particle to the cell that contains its position. This allows for quick access to the particles within a given neighborhood of a target particle, without the need to search the entire set of particles.

We consider a 2 dimensional example for illustrating the neighborhood search algorithm. We consider that each particle is considered to have an unique index $i[1, N]$ where N is the number of particles used for the simulation. We now create a neighborhood algorithm that creates index information of all particles $B_{k,mn}$ (a contiguous array) in the cell (m,n), i.e. $kN_{m,n}$ where N_{mn} is the total number of particles in the cell (m, n). Then we can calculate the cell by dividing the position of the particle by the cell size (we assumed the cell size as b here). For example, the particle i is stationed at cell $m_i = [x_i/b]$ and $n_i = [y_i/b]$. On one side, $B_{k,mn}$ is stored as a single contiguous array and now we need to access individual cells. So, we declare another array that is equal to the number of cells. Now, the neighborhood search will compute cell m_i, n_i for particle i and then loop over all other particles in cells $(m_i+ = 1, n_i), (m_i, n_i+ = 1)$and $(m_i+ = 1, n_i+ = 1)$. For a 2D, it looks for 9 cells where as in 3D, it loops over 27 cells. [3]

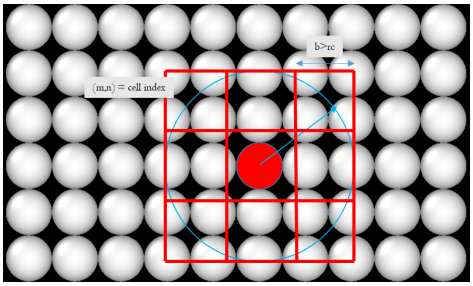


Figure 2.1: Data structure used in spatial grid decomposition

1. Created domains with length(l) > rc and used indexing (m,n) = ([x/l],[y/l]).
2. Assigned cells to all the particles using the formula x + xnum(0) * (y + xnum(1) * z) where xnum is a Eigen::Array3X with values of number of cells in each direction.
3. Then we store the cell information of each atom in the variable called Eigen::ArrayXi atomtocell and store the sorted array info in Eigen::ArrayXi sortedatomindices
4. Then we create a hash map to store the data of each particle and as key and the neighbors as its value to the hash class Neighborlist.
5. This returns a list of particles and another list with its neighbours.

[3]

3

Implementation

In this chapter, we discuss about the algorithms we used for developing kernel and it's tests. We also talk about the complete flowchart of how the solver has been designed and present some critical pseudo codes for that.

3.1 Kernel

In this section, we refer back to the section 2.1 for the numeric to be solved where our task is to develop two functions of kernel - one is the kernel itself and its first derivative that decides the gradient.

Algorithm 1 Kernel Algorithm

```
1: procedure KERNEL(i, j)
2:     for all particles do
3:         dist = normalized distance i-j
4:         α = 5/(14πh²)
5:         calculate q = ||r||/h
6:         calculate weight for three cases of q 2.1
7:     end for
8:     return kernelweight
9: end procedure
```

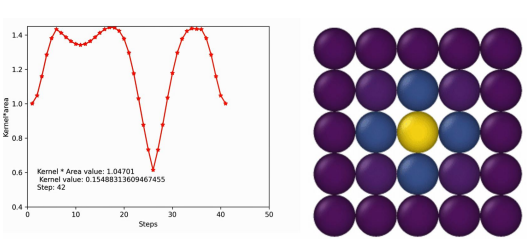
Here, the function $KERNEL$ take in two particle positions as parameters and returns the weight of particle j to particle i.

3.1.1 Test

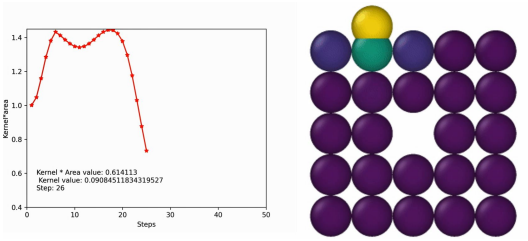
One way to test the function is to print the total kernel values for each particle and observe them with hand calculations. Also, we can observe the trend for three cases - when the particle is surrounded by exact number of particles that we use to calculate the density (ideal case), when the particle is surrounded by incomplete neighborhood and when it has more particles than ideal. We expect the ideal case to yield the summation as unity because of the kernel property mentioned below[2]

$$\int_{\Omega} W(x' - x, 2h)dx' = 1 \tag{3.1}$$

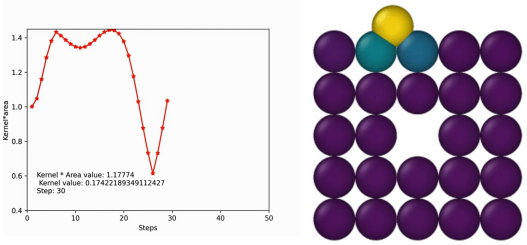
Also, one test would be to check if ∇W and $\nabla^2 W$ exists. The below table presents the three cases of a particle having less, exact and more particles around it and how the kernel



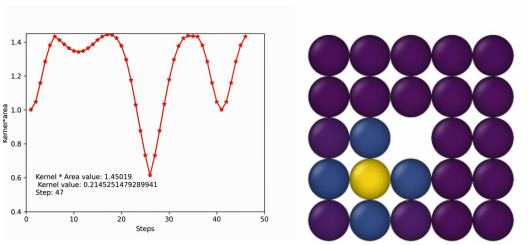
(a) Exact boundary ($W \cdot A = 1$)



(b) Particle with less density $W \cdot A < 1$



(c) Particle close to to particle $W \cdot A = 1$



(d) Particle with more density $W \cdot A > 1$

Figure 3.1: Snapshots of weight distribution of particle i at different positions w.r.t its neighbor. Color helps to realize the relationship trend of weight value that is dependent on distance between particle i and j. Here, the values plotted are $W \cdot A$

is influenced. The figure **??** helps to observe how the kernel values of a particle change when the neighborhood changes.

Here we write the analytical solution for the kernel function. 1.

$$W(0) = \alpha = \frac{5}{14\pi h^2}((2-0)^3 - 4(1-0)^3) = \frac{20}{14\pi h^2}$$
$$W(1) = \frac{5}{14\pi h^2}(2-1)^3 = \frac{5}{14\pi h^2}$$
$$W(\sqrt{2}) = \frac{5}{14\pi h^2}(2-\sqrt{2})^3 = \frac{1.005}{14\pi h^2}$$

Thus, we can from here calculate the cumulative summation of all the kernel weights.

$$\Sigma_j W_{ij} = W(0) + 4W(1) + 4W(\sqrt{2}) \approx \frac{1.001}{h^2} \quad (3.2)$$

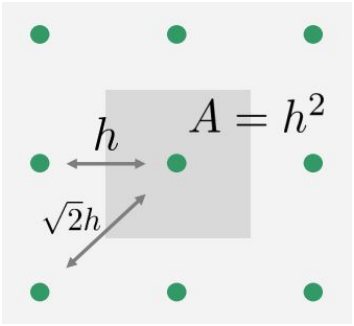


Figure 3.2: Grid description for test

3.2 Kernel Derivative

In this section, we present the algorithm used for kernel derivative (1st order) and the relevant test to check the functionality. We need to have the derivative to interpolate physical quantities such as density, pressure, and velocity from the particle positions. This allows for a smooth representation of these quantities, which is crucial for the accuracy and stability of the simulation. Also, for gradient estimation to calculate physical forces like pressure forces, viscous forces e.t.c. We use the same algorithm of kernel function here just by altering the weight equations for specific q.[2]

3.2.1 Test

We solve the 1st derivative manually by hand to see the expected matrix to be formed. Since it is a gradient and is a vector of 2 coordinates, we have a 2D matrix as the outcome. We

verify this by calculating the outer product of the particles.

$$\Sigma_j(x_i - x_j) \otimes \nabla W_{ij} = -\frac{1}{V_i} \cdot I \tag{3.3}$$

There is one more check to this where the summation of kernel derivatives yield zero. i.e. $\Sigma_j \nabla W_{ij}$=0

Here we write the analytical solution for the kernel outer product which is a 2D matrix. We here refer to the 1st order derivative 2.4 for the solution.

$$\begin{aligned}
\nabla W((0,0)-(0,0)) &= (0,0) \\
\nabla W((0,0)-(h,0)) &= (\frac{3\alpha}{h}, 0) \\
\nabla W((0,0)-(0,h)) &= (0, \frac{3\alpha}{h}) \\
\nabla W((0,0)-(h,h)) &= (-\frac{1}{h\sqrt{2}}\alpha\beta, -\frac{1}{h\sqrt{2}}\alpha\beta) \\
here, \beta &= (-3)(2-\sqrt{2})^2
\end{aligned}$$

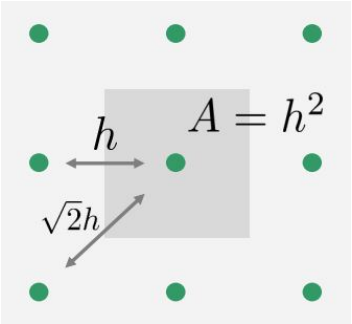


Figure 3.3: Grid description for test

Similarly we perform the calculations for the other neighbors and can conclude the following,

$$(x_i - x_j)_x \cdot (\nabla W_{ij})_y = (x_i - x_j)_y \cdot (\nabla W_{ij})_x = 0+0+0+0+0+\frac{1}{\sqrt{2}}\alpha\beta - \frac{1}{\sqrt{2}}\alpha\beta + \frac{1}{\sqrt{2}}\alpha\beta - \frac{1}{\sqrt{2}}\alpha\beta = 0 \tag{3.4}$$

$$(x_i - x_j)_x \cdot (\nabla W_{ij})_x = 0+-3\alpha+-3\alpha+0+0+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta+\frac{1}{\sqrt{2}}\alpha\beta = -6\alpha+4\frac{1}{\sqrt{2}}\alpha\beta \tag{3.5}$$

Thus, we have $\Sigma_j(x_i - x_j) \otimes \nabla W_{ij} = -\frac{1}{V_i} \cdot I = \frac{1.013}{h^2}$.

3.3 Boundary Handling

Boundaries are crucial part in simulation. In multi-physics simulation, we often perform simulations to observe state transformation due to rough contacts with boundaries. Also, to perform a meaningful simulation, we need to have an algorithm for handling the boundary. The main idea behind the boundaries that is being used here is particles with high pressure acceleration. We create two layer of solid particles (represented as boundaries) and when the fluid particles move collide with the dry nodes, the density of the fluid particles increase thereby increasing the pressure of the moving particles. This creates an additional force in the opposite direction. This is decided by stiffness constant as K is the constant that relates density with pressure. Higher the K, more is the effect of boundaries onto the fluid nodes.[2]

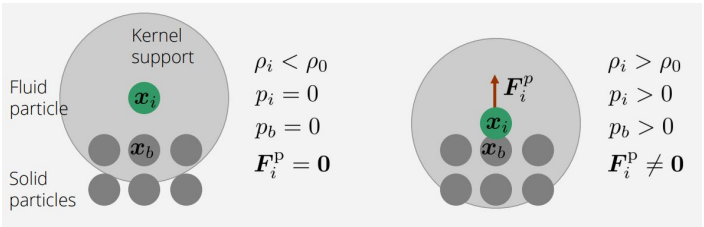


Figure 3.4: Schematic of boundary creation in terms of increase in pressure.

An updated understanding on how the interactions work would be to separate the wet and dry particles. So, the density and the pressure computation would change like,

$$\begin{aligned}\rho_i = \Sigma_{i_f} m_{i_f} W_{i_f} + \Sigma_{i_b} m_{i_b} m_{ib} W_{ii_b} \\ m_i = m_{i_f} = m_{i_b} \\ p_i = k(\frac{\rho_i}{\rho_0} - 1)\end{aligned}$$

When the fluid particles come close to dry particles, due to the kernel effect, ρ_i increases thereby increasing the pressure p_i. The contribution of the pressure acceleration is written as,

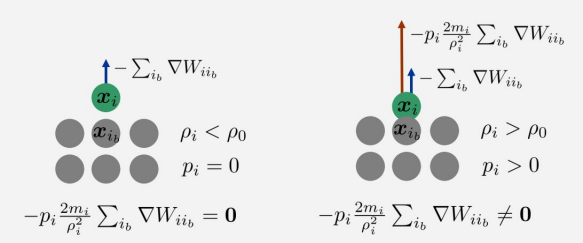


Figure 3.5: Detailed pressure and density variation with fluid-boundary interaction

$$a_i^p = -m_i \Sigma_{i_f} (\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2}) \nabla W_{ii_f} - m_i \Sigma_{i_b} (\frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2}) \nabla W_{ii_b}$$

Boundary pressures are unknown a concept named mirroring is used where we replicate the fluid pressure adjacent to the boundary.

3.4 Algorithm

Here an algorithm is presented that generates a simple fluid solver based on the pressure-valued boundary enforcement. We use the mentioned neighborhood algorithm 2.2 for creating the neighbor list.[2]

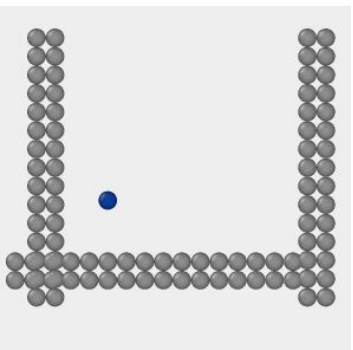
Algorithm 2 SPH Fluid solver algorithm

procedure SPH(*particlelist*)
  for `all particles` **do**
    find neighbors j
  end for
  for `all particles` **do**
    $\rho_i = \Sigma_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \Sigma_j m_j W_{ij}$ ▷ Compute Density
    $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ ▷ Compute Pressure
  end for
  for `all particles` **do**
    $a_i^{nonp} = \nu \nabla^2 v_i + g$ ▷ Compute non pressure acceleration
    $a_i^p = -\frac{1}{\rho_i} \nabla p_i$ ▷ Compute pressure acceleration
    $i(t) = i^{nonp} + i^p$
  end for
  for `all particles` **do**
    $v_i(t + \Delta t) = v_i(t) + \Delta t a_i(t)$
    $x_i(t + \Delta t) = x_i(t) + \Delta t v_i(+\Delta t)$
  end for
end procedure

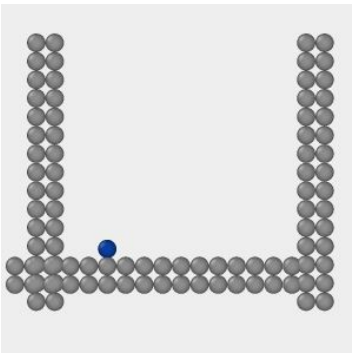
4

Results

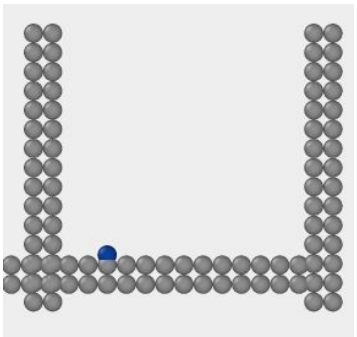
In this chapter we present two SPH simulations (one with a single particle and another with one million particles). Simulation snaps are presented here along with the setting descriptions like stiffness constant value, timestep value and viscosity value. In the figure 4.1, we can



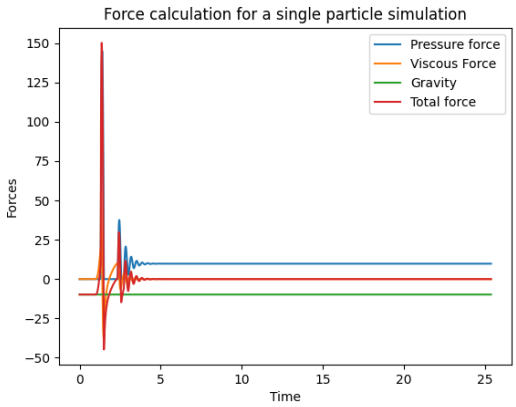
(a) at t = 0 ms



(b) at t = 0 ms



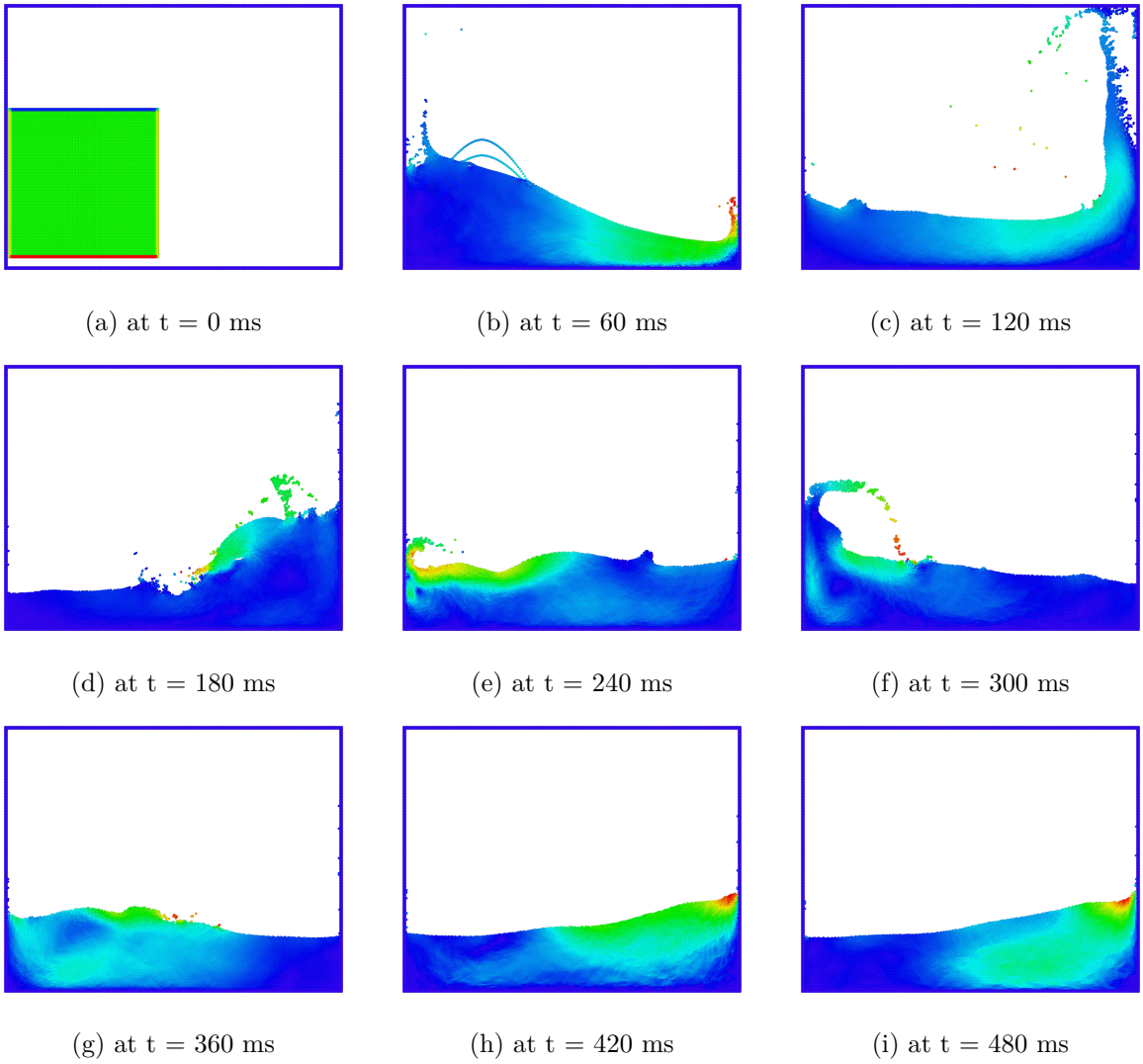
(c) at t = 0 ms



(d) Force vs time propagation

Figure 4.1: Propagation snaps of single particle simulation performed using smoothed particle hydrodynamics. Last figure shows the force vs timestep propagation

observe how the particle interact with the boundary. It is simulated keeping very low stiffness constant of 20. The timestep is kept at 0.001 seconds. The spikes that we observe is due to the initial boundary interaction of the fluid particle. The pressure acceleration increases when the fluid particle interacts with the boundary thereby causing the fluctuations.

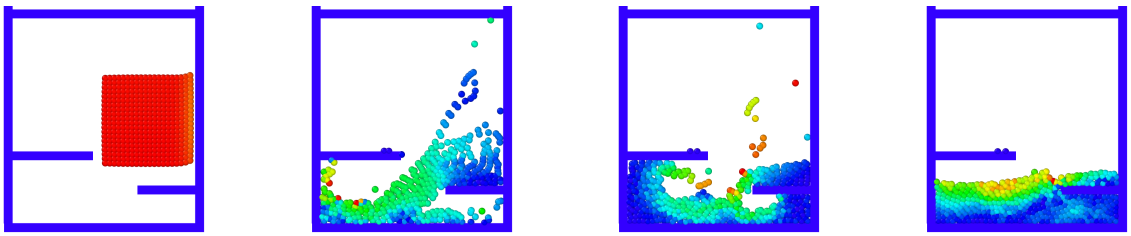


(a) at t = 0 ms (b) at t = 60 ms (c) at t = 120 ms

(d) at t = 180 ms (e) at t = 240 ms (f) at t = 300 ms

(g) at t = 360 ms (h) at t = 420 ms (i) at t = 480 ms

Figure 4.2: This figure shows the snapshots of a SPH simulation of 1 million particles. The stiffness constant is set to $K = 20000$, viscosity coefficient of $\nu = 8$ and timestep $\Delta t = 0.005$ seconds



(a) at t = 0 ms (b) at t = 20 ms (c) at t = 40 ms (d) at t = 60 ms

Figure 4.3: This figure shows the snapshots of a SPH simulation of 400 particles. The stiffness constant is set to $K = 30000$, viscosity coefficient of $\nu = 18$ and timestep $\Delta t = 0.005$ seconds

Here, we present some snapshots of a simulation where horizontal slabs are placed as boundaries and to see how the fluid particles interact with them.

5

Analysis

In this chapter, we focus more into analyzing the effects of the fluidic properties like viscosity, stiffness constant and also the effect of timesteps in the SPH simulation. Let us first assess the importance of timestep in SPH algorithm.

According to Courant-Friedrichs-Lewy (CFL) condition, the timestep of our SPH simulation has to be essentially small enough that the propagation of information through the fluid can be accurately resolved. It ensures the stability and prevents numerical instabilities. Also, this is because if we consider higher timesteps, the fluid particles might fail to interact with its boundaries between two consecutive timesteps. Timestep also decides the convergence of the simulation. Even though smaller timesteps will lead to accurate results (as the runge-kutta or explicit verlet/euler schemes are nothing but a prediction of discrete timesteps approximating a continuous curve, so small timesteps are much closer to the continuous profile), it can be computationally expensive and might have to run for a longer duration to achieve the steady state. [4]

Of course, we see increase in fluid pressure with increase in K. We performed a simulation on the stiffness constant and found that higher stiffness constant yields higher fluctuations in density deviation. This is mainly because the strong viscous force becomes less predominant making the fluid unstable. Also, the oscillation increases due to the fact that strong viscous forces can cause the fluid to oscillate more rapidly. Also, a mathematical interpretation would be to recall the formula $p_i = k(\frac{\rho_i}{\rho_0} - 1)$ and see if the K is increased, the density deviation is augmented to a higher value and since the viscosity is less predominant, adhesive property amongst fluid particles reduce thereby creating higher fluctuations in density deviation.

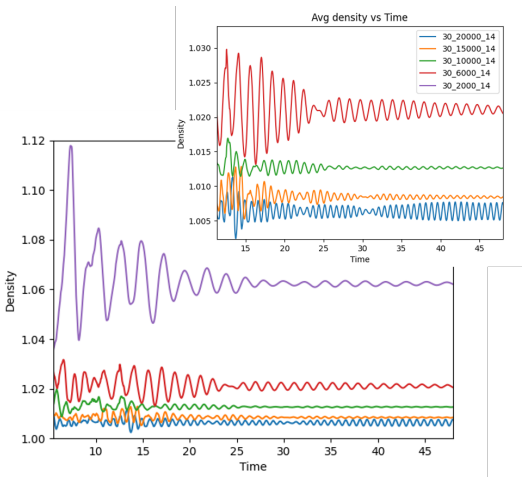
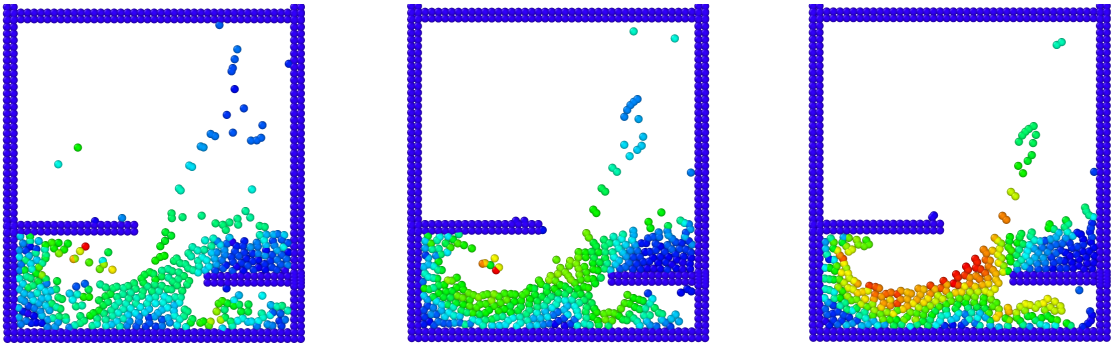
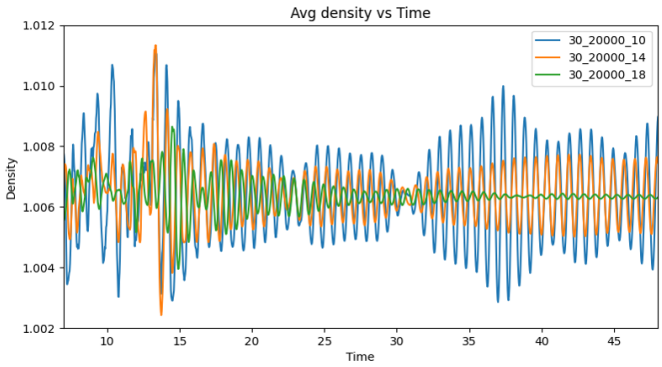


Figure 5.1: this fig presents how density deviation propagates with time for $K = 2000, 6000, 10000, 15000, 20000$ at $\nu = 14$ and $\Delta t = 0.03$ seconds

Now, we see how viscosity coefficient ν contributes to the simulation. By the definition of viscosity, we know that it is a property that restricts fluid movements by increasing the frictional values between fluid layers.



(a) at $\nu = 10$ (b) at $\nu = 14$ (c) at $\nu = 18$



(d) Average density vs time for $\nu = 10,14,18$

Figure 5.2: Plots (a),(b),(c) shows the particle movements for different viscosities and (d) presents a plot showing how the average density varies for different viscosity values

In this figure **??**, we see that for a sample instance of time, the particles with lesser viscosity moves more freely due to its less friction between layers. Also, in the last image we observe that the amplitude of oscillation decreases with increase in viscosity thereby providing stability to the fluid particles. But this happens when the stiffness constant is kept constant. When the stiffness constant increases, after a certain point the K becomes more dominant thereby increasing the fluctuations irrespective of the viscosity values. Thus, sensitive and clear differences in viscosity is observed when stiffness constant is kept small and vice-versa.

In this figure we observe the combinations that generates the profiles. Here, we tried using the minimum values of these three parameters to perform a feasible simulation. If the viscosity goes below the used values keeping the K constant, it explodes. Also, we can't take lesser k values because then it fails to create enough pressure in the boundaries to make it physically relevant. However, lesser values of K is possible if we reduce the timestep which can be computationally expensive. On the other hand, if stiffness is kept very high keeping the viscosity very low, it acts like an exploding gaseous phase with no fluid-fluid adhesion.

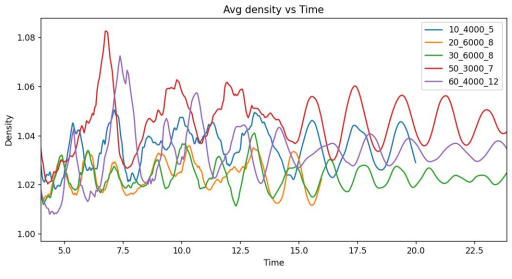


Figure 5.3: This figure plots average density for a combination of timesteps,viscosity coefficients and stiffness constant

6

Conclusion

In this report, we present a development of Smoothed particle hydrodynamics solver from the fundamentals to deployment on some use cases. The report concises the intricate details of SPH algorithm. Initially we discussed how we represent fluid as some cluster of particles (in lagrangian formulation) with some properties like position x_i, velocity v_i, mass m_i. Then an elaborate schematic is presented about to to search neighborhood particles efficiently with complexity $N \log(N)$. SPH is a discretization technique that relates one particle to another using some kernel functions. This has also been explained as how particles interact with each other.

We use cubic spline kernel and presented some mathematical expressions to represent the kernel function. Tests that are relevant for our simulation is also derived and shown. We simulated one SPH solver for a simple fluid dropping under gravity case. It is assumed that our simulation obeys Navier stokes equation and we presented an algorithm stating how the density, viscosity, pressure and total force is being calculated. Mathematical equations are also written in the form of an algorithm. In the next part, boundary handling has been presented that how we can use two layer of solid/dry nodes to represent boundaries that increase the pressure withing the fluid particles that come in close contact to the boundary layer. This trick is simply developed by exploiting the idea that we calculate density of a particle by taking the number of its neighbor particles into account.

Next to the conceptual part, we have discussed about how the algorithm is being implemented. We have spoken about the tests of kernel function and how the neighborhood algorithm is being implemented. This part is critical from development point of view as we need faster simulation outcomes for the results and analysis part. Moving forward, we have our Results chapter where the idea was to attach all the simulation snaps for different time spans. We performed a simulation with 1 million particles and observed its flow and compactness. Also, a simulation has been run on two horizontal boundaries that act as a dam for the dropping fluid. Finally in the analysis section, the main idea was to understand how stiffness constant, viscosity and timestep contributes to the simulation. Our main target should be to increase the timestep as much as possible respecting the Euler discretization laws. Also, higher viscosity yields very stable simulation, but we wanted to get a feel about how the viscosity and stiffness relates with each other.

This report is based on a course named "Laboratory in the research field "Graphics Data Processing"conducted by Prof. Matthias Teschner at the Informatik department of University of Freiburg. The solver algorithm is from his course module available under the course "Simulation in computer graphics" and a major part of work in this report is guided by him.

Bibliography

[1] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. A survey on sph methods in computer graphics. In *Computer graphics forum*, volume 41, pages 737–760. Wiley Online Library, 2022.

[2] Matthias Teschner. Simulation in computer graphics.

[3] Lucas Frérot Lars Pastewka, Wolfram Nöhring. Molecular dynamics, 2021.

[4] Carlos A De Moura and Carlos S Kubrusly. The courant–friedrichs–lewy (cfl) condition. *AMC*, 10(12), 2013.