University of Derby
Department of Electronics, Computing and Mathematics

A project completed as part of the requirements for
BSc (Hons) Computer Games Programming

Entitled

Real-time Simulation of Compressible Flow Using a Smooth Particle Approach

By
Jacob Green

April 2017

Abstract

Fluid simulation is a complex research area with a wide level of applications and studies in the field. Ranging from the highest level of Fluid Mechanical Physics to Fluid Dynamics, We will be researching the application of compressibility in fluid simulations and exploring the challenges associated with its implementation.

The simulation methods and approaches vary between physical accuracy and visual believability depending entirely on the type of application. Applications that focus on physical accuracy of a fluid would invest on more complex simulation methods that utilize more accurate real world physical laws to return a greater deal of fluid accuracy and behaviour, while simulations that don't require a focus on simulation accuracy and instead on visual believability will choose techniques that are more desirable for speed and visual properties, these simulations may choose to limit or even omit realistic rules to govern the fluid behaviour to further the desired behaviour.

This paper will explore traditional methods and factors that go into developing fluid simulations so as to learn more about modelling compressible behaviour in simulating fluids. I intend to explore the challenges in implementing compressible behaviour of fluids so as to learn about how simple compressibility effects can be implemented or modelled in simulations. The ultimate goal is to understand and identify challenges in developing compressible behaviour in soft-body fluid simulations.

Acknowledgements

Contents

Chapter 1

Introduction

1.1 Project Rational

Simulating fluid behaviour is a highly active area of research and has had many forms of investigation. There is a constant desire to simulate fluid for accuracy of physical models and visually for visual effects in different media.

A lot of modern day fluid simulation methods work with the conservation of volume, E.G Incompressible Fluid; the fluid system volumetric space can not change as it is physically incompressible. Common methods work by evaluating the system to balance a constant such as density. With technology constantly improving more and more computationally expensive real-time physics simulations are becoming more viable, I believe there is a place or will be a place for real-time compressible fluid simulations and I want to learn more compressible fluid simulations.

While we investigate existing simulation methods and try to develop our own method of simply simulating compressible fluid behaviour. We hope to learn and understand the complexity and challenges related to compressibility and how it could be applied in fluid simulations.

1.2 Project Aims and Objectives

The main objective of this paper is to research existing simulation methods for fluids, to learn about the different types of approach to handling fluid behaviour and the complexity of implementing compressible fluid simulation. With the ultimate objective of implement some form of compressible behaviour into a real-time dynamic fluid simulation.

Researching into the simulation of fluids to discover how they function; by learning how fluid simulations work we can discover differences in implementing compressible physics behaviour in a simulation.

By learning and exploring existing fluid simulations we can understand what would be needed to simulate fluid with a compressible behaviour.

Along side my research, I will attempt to implement a simple solution that can correctly incorporate some elements of compressible behaviour in a dynamic fluid simulation.

1.3 What Is a Fluid

A fluid is an substance with a physical state that does not have a fixed shape and that can have its shape changed by external forces. A fluid generally can only be in a liquid or gaseous physical state. A fluid can continuously deform when when in contact with a tangential force or sheer stress.

Fluids exhibit a non linear motion where the system can be moving in a multitude of different directions simultaneously.

Fluids can exhibit flow, which is when a particle of the fluid can freely move (not constrained) through the fluid structure from one position to another.

Fluid simulation systems regardless of compressible or incompressible flow will conserve the fluid systems mass.

Traditionally compressible fluid simulation is less commonly used in popular fluid simulation methods, as common fluids like water are by nature weakly compressible. Most methods would use incompressible models for fluids to handle common fluid cases. You are more likely to find compressible fluid simulations used in Physical simulations which prioritise accuracy, such as aero-dynamic air flow simulations, where at high speeds the fluid can experience high compression.

1.4 What is Fluid Dynamics

Fluid dynamics is a field of study into calculating the physical behaviour and properties of fluids with regards to movement. This includes handling for the flow velocity, pressure of the fluid, density of the fluid and temperature of the fluid.

1.4.1 Laminar Flow and Turbulent Flow

Laminar flow and turbulent flow are two different types of fluid flow behaviours.

Laminar flow is also referred to as a stream line flow it is where you would see a parting of flow when approaching an obstacle. It is more apparent at low velocities.

Turbulent flow is the more commonly seen Fluid flow behaviour in real world physics. Turbulent flow is more chaotic and unpredictable in behaviour, exhibiting vortexes and curling of the fluid flow within its fluid. This is caused by excess energy existing to overcome the viscosity limitations of the fluid (more common in low viscous fluids).

1.4.2 Viscosity

Viscosity is a commonly known term to describe the ease of movement for different types of fluids. A fluid that has a high viscosity will generally be more resistant to deformation of shape and volume, while a fluid with a low viscosity will more easily move and be deformed.

A common fluid viscosity comparison is that of Water and Honey; with Water having a low viscosity and Honey having a high viscosity.

1.4.3 Why Fluid Dynamics Is Important

Fluid dynamics is a mathematical field of simulating the real world physical properties of fluids. How it moves and interacts physically in the world. It is crucial in many models for fluid behaviour.

Fluid dynamics is useful in many real world domains, extending from mechanically physical accurate simulations to real-time applications that are more dependent on believability that accuracy. It is used extensively in engineering for accurately modelling the behaviour and physical effects of fluids involved in engineering projects. It is also used for simulating fluid behaviour visually, mostly in rendering to get fluid properties to act like a actual fluid.

It is used frequently in scientific models for things like aviation and mechanics. An example would be using a fluid simulation for simulating the effects of aero-dynamics for an object. Real-time visual effects usage would be for entertainment in the Computer Generated Imagery (CGI) discipline or for video games.

1.4.4 Incompressible Flow

Incompressible flow works by resolving differences in the distribution of fluid volume in a space. The fluid can propagate through space with the fluid mass moving to different regions of the fluid.

The fluid should not exhibit any change of density as it undergoes flow. Incompressible fluids do not change their volume much[Cui, 2009].

1.4.5 Compressible Flow

The difficulty with compressible flow is that the system has to balance the change of volume and exhibit appropriate force into the system, with out invalidating fluid dynamic rules.

Compressible fluids change their volume significantly [Cui, 2009], as it experiences compression of mass into a smaller volume.

1.4.6 Density and Pressure

Density is the amount of mass contained in a unit of volume and Pressure is the amount of force per area.

In traditional physics, the pressure of a gas is calculated as $P = \rho RT$ where P is pressure, ρ is density, R is the specific gas constant and T is temperature of the gas.

In traditional physics, the pressure of a liquid is calculated as $P = \rho Gh$ where ρ is fluid density, G is the gravity acceleration constant and h is the fluid depth.

For handling fluid behaviour we can derive that Pressure is directly proportional to the value of density.

Chapter 2

Literature Review

2.1 Introduction to Fluid Simulation

Fluid simulation methods must handle some defining characteristic properties of fluids. The non-linear motion that a fluid exhibits at different spatial points in space and time. The conservation of density and mass of the fluid system.

A fluid can have velocity acting in more than one direction simultaneously, with the energy of this motion being seen as flow; distribution of the volume and density. Other properties like vorticity and flow are introduced to the simulation in a variation of methods. A fluid is said to be incompressible when the density of fluid at all points and times is constant.

Predominately there are only two types of mathematical approaches to solving these properties; Euler and Lagrangian methods.

2.2 Euler Grids

The Euler method works by spatially dividing the fluid system into a spatial grid and resolving the cells of that grid, such that the cell properties can be used to dictate the simulation behaviour from cell properties such as the fluid distribution.

The grid needs to keep track of at least two required pieces of information the fluid velocity and the fluid density[West, 2008]. Different Euler based methods may have additional properties to manage, such as for effecting flow.

The simulation revolves around propagation, calculating when the fluid would cross the grid boundary's and applying forces to neighbouring cells adjacent to the boundary. This forms a complex network of cells that propagate the fluid properties to neighbouring cells.

Euler grid methods require a fixed region of space to operate in as the spatially partitioned regions function on a boundary system; the grid cannot have it's cells adjust in runtime as this would distort the fluid properties, the cell would have a differing volume for the fluid, causing a difference in density and

actively generating fluid movement along the cell boundary. This would indirectly generate fluid flow and simulation inaccuracies.

2.3 Lagrangian Particles

The Lagrangian method is a particle approach. It is more dynamic and not space restricted.

To calculate the fluid density at a point in space, the method of calculating the density needs to be independent of the absolute position of particles, relying on the relative separation of the point masses to extract a value of density[Price, 2011]. The method must also be time independent in respect to the particles [Price, 2011], avoid evaluating the change of position or energy on the particles.

In simpler terms Lagrangian calculates the density at a spatial point by observing surrounding particle point masses. The calculation is attributed to a kernel weighted sum[Price, 2011].

These methods analyse the surrounds to established a probability for the distribution of pressure to calculate that particles behaviour Other Lagrangian particle based models such as a Probability Distribution method of Smooth Particle Hydrodynamics[Welton and Pope, 1997] or a "Lagrangian PDF Method for turbulent flows[Pope, 1994] also utilise a kernel weight to extract density of the system..

Particle approach should have it's properties be fully conservative with out the need of external additive artificial dissipation/diffusion of energy [Price, 2011] or removal from handling in the system.

Some key differences and similarities between these two methods is that the grid approach resolves density via the volume of each grid cell and the particle approach resolves the density via inspection of surrounding independent particle point masses.

2.4 Things to consider for real-time fluid simulation

Technology is constantly getting better, computationally expensive simulations are becoming more viable to simulate in real-time There are still a few key things to consider for real-time simulation.

Computational cost for real-time [Bridson and Müller-Fischer, 2007], it has to be maintainable at a update rate consistent for the application. Usually 40-60fps for video game usage [Müller-Fischer,].

Memory cost considerations if being used in real-time. Memory is more limited in real-time applications as it is shared between other processes. The method for real-time simulation must function with less available memory [Müller-Fischer,] [Bridson and Müller-Fischer, 2007].

Simulation stability, real-time simulations can not use correctional frame-rates like off-line simulations can[Bridson and Müller-Fischer, 2007][Müller-Fischer,].

Believability of the simulation, if a simulation has to be reduced in complexity to suit the requirements of the real-time application. The intention is that the fluid simulation is still plausible and real[Bridson and Müller-Fischer, 2007].It needs to get real-time behaviour as close to off-line simulated behaviour as possible [Müller-Fischer,].

2.5 Navier- Stokes and Fluids

Navier- Stokes is a well established set of formulaic rules for solving fluid dynamics. It presents a natural framework for fluid modelling[Stam, 1999]. It has seen use across many different methods including the popular Stam's method[Stam, 2003].

There are a few key physical laws that need to be considered along side the Navier-Stokes equations for flow. First being known as the 'conservation of mass' formula or as the continuity equation. The fluid mass conservation law basically states that for a fluid moving at a constant speed the amount of mass entering a system is equal to the mass leaving a system.

Simulations relying on the Navier-Stokes equations can use the Navier-Stokes continuity equation, in conjunction with Navier-Stokes velocity calculation to solve incompressible-flow simulations[Stam, 2003].

Navier-Stokes form of vector continuity equation is used in handling conservation of linear momentum. The change of density over the change of time plus the divergence of density over the velocity vector fieldu is equal to zero.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1}$$

The continuity equation(1) can be further derived for incompressible flow(2) such that $\nabla \cdot u = 0$, results in no change of divergence of the velocity field; with $p = mv$, zero change of velocity in the momentum can prove conservation of mass.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot u = 0 \tag{2}$$

Navier-Stokes is not fully solved it is noted that an area of complexity exists within the Navier-Stokes equations; the problem of whether there is the presence of a smooth solution that correctly solves the Navier-Stokes equations. Most simulations have additional handling and rules set up to handle non-smooth cases, these are seen as vorticity handling and artificial dissipation of force.

Another valuable Navier-Stokes equation is the flow momentum equations(3). The momentum equation(3) can be evaluated to a change in velocity is equal to viscosity k times diffusion of the fluid, minus the velocity divergence of the velocity $(u \cdot \nabla)u$, minus pressure gradient $\nabla \rho$, plus external forces f. [Cui, 2009].

$$u = K\nabla^2 u - (u \cdot \nabla)u - \nabla \rho + f \tag{3}$$

For compressible flow we will replace the divergence velocity value with that of the compressible mass continuity equation $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u})$.

2.6 Particle Simulations General Concept

Lagrangian and Euler models are the mathematical approaches to handling the simulation of the non-linear physical behaviour of the fluid.

A particle simulation method functions by being able to evaluate the fluid at any point in space based on and inspection of the fluid system and surroundings. Functionally the simulations break down the properties of fluid into a collection of particles which defines the behaviour of fluid at the point space of the particles.

A particle would be a single representation of a fluids mass. The particle would exhibit other properties that traditional physics particles would have, velocity, mass.

2.7 Impact Shock-waves

A factor of compressible fluid is the ability for the fluid to absorb force in shocks and dissipate the energy absorbed in the fluid. Lind, SJ covers a method of simulating wave slam via two phase compressible and incompressible Smooth Particle Hydrodynamics to handle the shock [Lind et al., 2015].

2.8 Boundary Conditions

There are different types of boundary conditions that can be considered in simulations. Boundary conditions are needed for handling the fluid behaviour when interacting with an external environment or system.

A boundary between physical systems is representative of what happens when surfaces become in contact; they generally experience a frictional contact force reducing the movement at the boundary location.

Not handling the behaviour, can cause issues such as irregular properties of the fluid when interacting at a boundary.

2.8.1 No-slip Boundary Condition

In the real world a boundary would be the behaviour of fluid in contact to the surface of an external system such as a container. A no-slip boundary is used to approximate the behaviour of fluid when meeting a physical surface.

A no-slip boundary occurs for viscous flow and all fluids should have a finite viscosity. The boundary should impart its velocity on the fluid at the boundary.

A no-slip boundary operates with the behaviour that fluids at a boundary should exhibit zero velocity relative to the velocity that the surface is on. Particles near the boundary should experience a lower velocity nearer the boundary

surface than fluid free of surface contact. This is because in traditional physics you would expect a contact force such as friction to reduce the velocity motion.

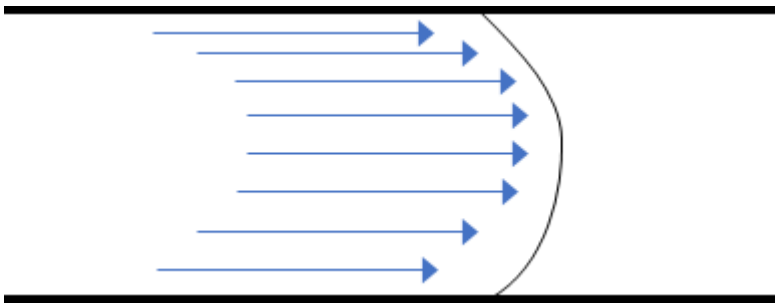

Figure 1: Image of fluid flow through a pipe. Fluid at the boundary surface are slowed by the contact surface via Viscosity

A no-slip boundary condition will simulate the effect of a fluid moving along the contact, what should happen is the fluid would move slower the closer it is to the boundary surface, using a no-slip prevents it from moving unrestricted and instead makes the fluid move away from the surface.

2.8.2 Capillary Boundary Condition

A capillary boundary is used to handle the behaviour of a fluid surface when two different fluids would meet in interaction.

At a Capillary Boundary something thrown into the boundary should take a specific velocity through the boundary[sta, 2015]. This implies the adoption of the boundary velocity.

2.8.3 Reflection Boundary

This boundary collision is another type of boundary condition that can be used for simulating the behaviour between a fluid and a physical domain. It is used to simulate inelastic collision with the boundary. A restitution value is used to control the dampening of the fluid particles that hit the boundary. It dampens the velocity of particles near the boundary while redirecting the movement to be reflected from the barrier.[Bindel, 2011]

2.9 Simulation Models

2.9.1 Euler Grid - Advection Method

Advection which is the process of propagation, this would be the motion of fluid matter or heat in one cell being transferred to a neighbouring cell, propagating the fluid. This propagation of heat or matter across spatial cells is caused by the fluid flow and is a trait of the flow.

Forward Euler Advection step is unstable when used with discretized spatial region [Bridson and Müller-Fischer, 2007].

Divergence (want to keep it low for stable simulation) which is the measure of fluid entering verses leaving a single cell in the grid. A very stable simulation should have an equal amount of a fluid entering and leaving, due to propagating flow of the fluid. A divergence where there is an unequal balance of fluid leaving a cell compared with fluid entering a cell is a potential sign of instability, because only so much fluid can be evacuating from that space with out being replenished in the system.

Solving pressure can be done by calculating a pressure gradient and applying it to particles that occupy the discretized spatial cell of our Euler grid.

This method is only functional in a fixed region, the grid defines a volume that the fluid can be simulated in. Moving the defined spatial grid is not trivially easy either because transforming its location could disrupt the boundary condition.

Dissipation is a backwards stepping approach where a average of values from the previous iteration time step is removed from the previous time steps force [Bridson and Müller-Fischer, 2007]. What it basically does is continually take an average away smoothing out the behaviour. It is usually used in Advection methods though it is advised to avoid them as they can be unstable.

Applications using this simulation concept could utilise this method with additional accuracy by further breaking down the grid into more refined resolution of pressure, as well for refining the grid around complex regions of space for defining boundary conditions, especially boundary conditions around external interacting system such as rigid physical geometry and other fluid simulations.

2.9.2 Jos Stams Stable Fluid Method

Jos Stams Stable Fluid [Stam, 1999] method is a popular method that is used for modelling incompressible flow. Its foundations are based on Euler Grid mathematics style of resolving the density, processing diffusion in the grid system. It uses the Navier-Stokes equation for modelling the flow of fluid in the grid system. Looking back on the Navier-Stokes continuity equation used for incompressible flow, to be accurate this approach requires conservation of mass, to do this the simulation artificially removes divergence in the system to guarantee mass conservation.

Stams Stable fluid approach is better suited in the 2D domain.

The exact steps for Stams method is to create a discretized grid of densities. Then the grid receives 3 unique update steps. The first step will be the source density, this can be provided initially by the grid. Multiply the initial grid by the time-step and add it to the density grid. Then you have to apply the second step which is the diffusion step, this is handled by calculating the density flux for each face of the spatial cell multiplied by the time step. The final step is to move the fluid by time stepping the velocity of the fluid. Step two and three can be unstable if the diffusion rate is too high, discretized spatial grid is too

fine (small), or time step too large, because the density propagates across the cell into the next neighbour cell.[Stam, 1999][Stam, 2003].

2.9.3 Smooth Particle Hydrodynamics (SPH)

A popular fluid simulation method is "Smoothed-particle Hydrodynamics" (SPH), originally designed for liquid fluid simulation it is a popular method that has seen variations in the past. SPH is a fluid simulation method that is primarily used for handling the behaviour of water(liquid).

SPH works by simulating many point mass particles in a simulation, each of them have properties which are used when solving the simulation. Particles represent fluid volume, a representation of the fluid mass at the spatial point. They also have properties for resolving motion in the system, such as velocity and acceleration.

SPH is a Lagrangian fluid simulation method. Surrounding particles around a particles position are used to calculate a density distribution. This is expensive with a complexity order of $O(n^2)$, some form of optimization is commonly used to limit the amount of surrounding particles that need to be queried to calculate the density distribution. Methods of doing this are by by limiting the query distance of particles H[Müller-Fischer,] and/or caching of existing particles within a spatial discretized system(spatial hashing)[Müller-Fischer,] to make retrieval of neighbouring particles easier.

This density gets utilised by a a smoothing function known as kernel function to handle the separation of particles. Inspect the properties of neighbouring particles to retroactively obtain the physical qualities of the fluid at that particular point in space.

Another approach to fluid simulation are methods that rely on a probability distribution to get a non-deterministic believable result.

Many Lagrangian based probability distribution methods implement Monte-Carlo numerically[Pope, 1994].

2.9.4 Related Work on Compressible Smooth Particle Hydrodynamics

[Becker and Teschner, 2007]

A numerical prediction of water-air wave slam using incompressible-compressible smoothed particle hydrodynamics[Lind et al., 2015] handles compressibility by using two phase flow with one phase of the fluid being compressible and the other incompressible. The air phase is the one that is compressible and it is done via a weakly compressible SPH method, while the water in the paper is using an incompressible model.

[Amanifard and Namini, 2012]

A probability density function (PDF) incorporated with techniques from SPH proposed by Welton, Walter C [Welton and Pope, 1997], can be used to derived the mean pressure of a particle. The method is entirely Lagrangian based with techniques to calculate the pressure field from the particles. This

PDF function uses a Lagrangian mass density function to calculate the mass density for a particle. This can be used to model compressible fluid. However this method isn't suitable for handling shock impacts.

2.9.5 What is compressibility

Compressibility of the fluid effects how much mass can inhabit a volume and how strongly it can change the momentum of the particle, higher regions of pressure exerts a greater change of momentum on the fluid.

From a basic perspective simple approaches of getting compressible behaviour are dependent on measuring a change of density in localized spatial regions and then applying some form of numerical calculation to resolve differences in density appropriately.

2.9.6 The Trouble With Compressibility

There are different factors to take into consideration when simulating compressibility. The method of discretizing the density; to generate a better representation of the pressure distribution in a fluid system. The method of resolving the compression such that the pressure can return to a balanced pressure state, all while still maintaining other fluid properties such as the fluid flow.

Chapter 3

Research Methodology

3.0.1 Introduction

I intend to investigate fluid compressional behaviour by developing a method based on a Smooth Particle approach of representing fluids. I will create some varying methods to try and simulate compressible behaviour via resolution of discretized mass density.

I will research to see the difference in how compressible behaviour works under my different approaches of simulating compressible fluid behaviour. I will also be testing to see how my different methods perform in different simulated tests, including how the behaviour is effected under high complexity costs such as more particles to simulate.

There are a few different approaches that can be used to research and learn about my methods. Traditionally some aspects of compressible fluid simulations are analysed statistically due to the complex and chaotic nature of some aspects of compression, such as turbulent flow. My analysis will focus on measuring the net energy in a system to see if my approximate methods correctly restore the system back to a balanced equilibrium of pressure. To do this I will be performing a numerical analysis on the momentum in a fluid system and mathematical observation of the fluid state.

I will be evaluating whether my method can experience a form of compressionable behaviour such that it can attempt to resolve high density regions to balance our the pressure in the fluid system.

3.0.2 Research Strategy

My research will be oriented around creating a real-time simulation framework that will allow me to physically simulate my fluid methods, rendering of crucial information for improving development (pressure rendering, particle rendering, etc), plus provide a testing platform for me to monitor and record data.

I will record data on my experimental methods of handling compressible fluid, I will then perform an evaluation on that data.

While developing my methods of simulating compression I have referred to previously established material on compressible behaviour. To ensure that my methods exhibits some of the same properties and behaviour handling of other accepted methods.

Simulation Framework

I created a basic soft real-time interactive agent-based computer simulation; a term used to describe a semi-accurate model of a physics simulation, using approximations and simplifications in the simulation model [Gregory, 2009].

I choose this framework because it is better suited for creating and simulating real-time behaviour with the ability to be extended into a pre-computational model for higher simulation accuracy if needed. By using a semi-accurate physics model I can make some physical assumptions to avoid some complex areas of fluid physical behaviour.

Because the framework I have developed is targeted for real-time simulations, it will be easier to reiterate development of my methods due to being able to visually inspect the simulation and make some initial evaluations on the fluid behaviour. Being able to quickly evaluate and develop my simulation methods allows me to create and developer more responsive behaviour.

To make the real-time simulation framework I used C++ as the language of choice. I used OpenGL 4.3 and glew 2.0 for rendering. I used GLM an open source mathematics library that is well suited for use in rendering with OpenGL, this saved me from having to recreate a vector mathematics library.

I will be performing a variety of experiments with different modelled systems of handling pressure distribution of a fluid representation.

Collision detection is a commonly used approach in real-time collision engines to detect if geometry representation are intersecting.

Physics Evaluation

To evaluate my system I created a Euler Grid that covered my simulation space. The resolution accuracy is completely up for discretion. Though I chose to use a resolution of 1 unit as my cubic volume.

Each iteration of my simulation calculates where the point masses reside within the discretized spatial grid. Once all the grids have calculated their representing mass, the grid system evaluates the density of each cell volume which is done by dividing the mass by the volume of the cell it resides within.

After calculating the density of each cell I have a system that has a discretized represents of the distribution of density, from that density I can evaluate the pressure using the previously stated assumptions of pressure and density relationships such that pressure is proportional to the density. I calculate a pressure gradient by inspecting a scalar difference in the pressure of each cell to generate a change of pressure gradient.

Using this spatial grid I can evaluate concentration of mass, investigate density at spatial regions and evaluate the pressure gradient to see what direction

my fluid system pressure should be applying a momentum change to respective surrounding spaces of the fluid.

Compressible Spring Method

I started developing a method that could be very simply integrated into other real-time simulated physics frameworks. The approach being point masses with a integrated region of influence, a volume that the mass could represent.

I had a fluid simulation class that simulated the fluid, this simulator class had other global fluid properties such as the viscosity term used to control the fluid flow behaviour, a compressibility limit term, a scalar that dictates the maximum acceptable compression of the mass into a particles volumes and a restorative force term, a scalar for controlling the strength that particles repel.

Based on the Lagrangian Smooth Particle methods I create many point mass particles that have a position, property of Mass and a radius value that is used to define the spherical volume the particle mass represents.

Many fluid Lagrangian methods define a weighted kernel function [Welton and Pope, 1997] [Amanifard and Namini, 2012] [Becker and Teschner, 2007] [Price, 2011] that is used to calculate what neighbouring particles are calculated into creating a representation of the density distribution at a spatial point. This distribution is a form of discretizing the density of the fluid system into a spatial region.

In my method I chose to use collision detection to calculate which point mass particle volumes were undergoing any form of intersection, the intersection was used to detect which particles I should include for calculating a change of density for my particle. The benefit of using collision detection is being able to interface it with existing real-time collision detection systems, such as those used in modern real-time simulations engines.

With these particles representing a mass and influential spatial volume, any form of intersection would be a case where the mass in the particles respective spatial volume is increasing or vice versa the containing volume of the mass is shrinking, this results in an increase of density, proportionally increasing the pressure force a mass particle influences on neighbouring intersecting particles.

When multiple particles would begin intersecting the calculated increase of mass is handled based on the intersection depth of the opposing intersecting particle. The radius volume of the particle is used to convert the intersecting depth to a normalized scalar of magnitude to work out how much additional mass occupies a particle volume.

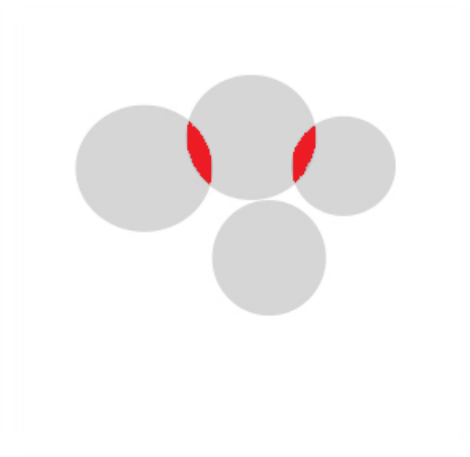

Figure 2: Depiction of overlapping Lagrangian Spring Particles. The interception depth marked in red is used to calculate the change of mass in the respective intersecting particles, this is used to scale the repelling force. The intersection depth scales the amount of mass calculated in the individual particle.

Pressure Distribution Grid Method

Using the existing work I used for my physics evaluation system I implemented a pressure distribution method of simulating fluid that works based on the Euler pressure discretized grid. However using this method is isolated from traditional frameworks in that it requires handling for how the fluid controlled by the pressure grid will process interactions with physical surface boundaries.

To handle the physical surface boundary I used my existing collision detection engine to calculate where the physical geometry would intersect the spatial pressure grid cells. The cells that do intersect with physical geometry will store some important information on the geometry necessary for resolving the boundary condition. The spatial cell will be marked as a boundary cell, will record the velocity of the physical geometry and the surface normal of the intersecting geometry.

I use this boundary cell to implement a no-slip boundary condition. Using the information stored within the boundary cell I can interpolate the velocity of the particle to that of the velocity of the boundary surface. When the particles are at the surface the velocity of the particle should be equal to the boundary velocity.

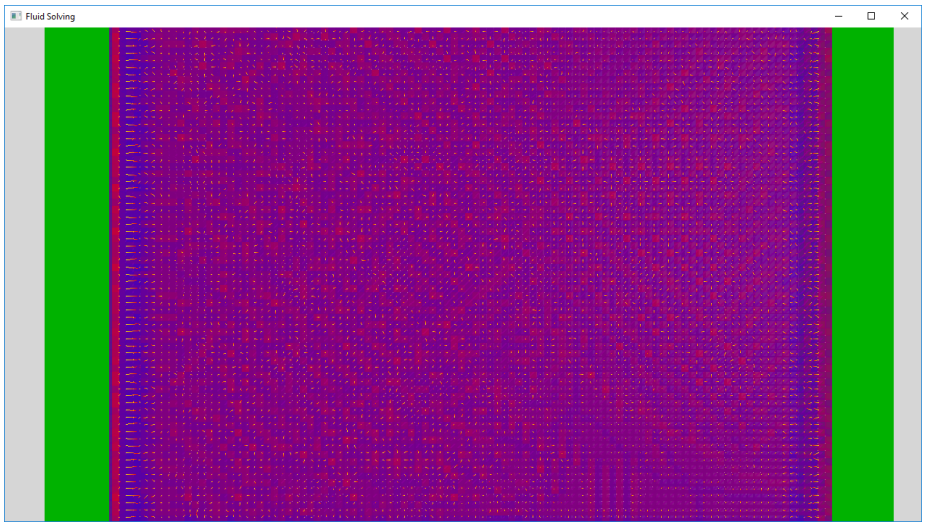

Figure 3: Depiction of Pressure gradients during a simulation, the high concentration of red means their is a higher mass density in that spatial cell.

3.0.3 Data Generation Methods

Create a numerical analysis of all the particles simulated in the methods to calculate the system behaviour over time, such as change of velocity from collisions.

I will record the sum momentum of all particles in my fluid system to ensure that the net momentum is equivalent to the sum of initial energy I supply to the system.

My first test will be to spawn a fluid inside an enclosed box while simulating with a variation of my method. I then intend to apply a fixed force impulse into the centre of my simulated fluid test. I will then record the net momentum of the fluid system to measure the behaviour. I expect to see the system eventually return to a uniform equilibrium or I would expect to see no net momentum change after the initial impulse force is injected into the simulation. I will run a variety of tests using this enclosed box (figure 26). Depending on what I am testing I will either test the concentration of particles in the grid volume or adjust the volume and particles proportionally to test for differences in behaviour from particle counts to generate a range of test data to evaluate the methods with different orders of complexity.

For testing the Spring Method in the enclosed box we intend to perform 3 tests to evaluate that the behaviour is identical at different particle counts. The first test will be limiting the box to a 25 by 25 internal volume with the impulse burst centred into this volume and radius calibrated for this volume. This will then scale up to a 50 by 50 box and then finally a 100 by 100 box, so we will

have a decent range of results for this method ranging from 625 particles, 2500 particles to 10000 particles.

For testing the pressure distribution grid method I will scale up the pressure, unlike with the spring tests where I maintained a fixed ratio of particles to dimension, I will be scaling up the particles held within a volume to test how it behaves with different magnitudes of pressure.

The other test I want to demonstrate is a flow test to see how my method behaves under flow. This will be done by having a fluid move with initial momentum and see how it interacts with an obstacle in its flow.

I create a pipe with a blocking top and bottom length, in the centre of the pipe is an obstructing static cylinder that will block the particles; the particles will have to flow around the object which is one of the fluid properties I want to test. The particles would be spawned into the pipe with an initial momentum. Figure 5 shows an initial pressure view of the set up for this test simulation.

I will visually monitor how the fluid particles behave flowing through the pipe, I will also do a comparison test on using a boundary condition verses without a boundary condition to quantify how effective my no-slip boundary is. I will also test how different interpolations of pressure effect the pressure flow. The main test though will be to see if momentum is conserved with my flow methods.

3.0.4 Data Analysis

I plan to use a numerical analytic technique to evaluate the behaviour of my methods. Each test will have generated a measurement of the momentum in the system for every single tick of my physics simulation. I will evaluate the change of momentum to see if it will return to a state of force equilibrium and if it does restore back to equilibrium I will measure to see how fast it restores its state.

The previously mentioned Navier-stokes equations can be used to back up my findings in behaviour for my compressible fluid.

For stability and accuracy of my simulation I will force my simulation to run at a fixed phase-time. Many real-time dynamic simulations support various phase-times and have facilities to accommodate changes to the simulation update time, including interpolative facilities for the physics. I have not implemented all of these facilities in my framework, so to limit any variations in my data recording I run my simulation at a fixed update time loop.

3.0.5 Conclusion

A lot of time was spent establishing a testing framework that correctly allowed me to handle collision detection, real-time rendering, an external interaction system, and a testing framework to record numerical accuracy for my testing.

I would look into using an existing established collision physics engine framework. Good open source engines such as the Bullet Physics Library[Coumans et al., 2013]

would use more efficient collision culling technique to reduce the complexity of collision detecting. It would also provide a solid foundation to extend off, to develop a fluid behaviour method.

The rendering framework used was my own framework developed using low-level OpenGL 4.3. It took some time and ultimately had performance limitations, using a higher level 3D rendering library would have saved valuable time and would make it easier to develop and control the simulation. A popular 3D rendering library would be OGRE 3D available under the MIT-Licence

My research testing method could have been drastically improved had I found and implemented a reliable existing control method to refer against in testing. This would have allowed me to compare and better judge the behaviour of my simulation approaches correctly to that of the existing reliably accurate method. I would still expect my solution to end up differing a bit in behaviour as I will experiment with handling the compressibility in my fluid simulation differently. However it would have shown me the behaviour when interacting with external physical structures and shown me the flow behaviour I should be trying to emulate.

Testing the performance and the net energy handling of my simple compressible fluid method approaches under varying levels of complexity is a good method of validating the behaviour of my fluid simulation method because the momentum of the system can show me how energy in the system is handled. When under compression you would expect there to be an increase of potential energy as the system attempts to restore to a balance.

It is important that the appropriate consideration is considered for numerical handling in a simulation. Working with small numerical units and other precision related calculations can easily result in invalid simulation scales or invalid numbers in calculations.

Chapter 4

Findings and Analysis

4.1 Analysis Introduction

I will compare the momentum of my systems and observations of my simulations to evaluate the compressible behaviour of my simulation. In theory a accurate method would be able to return to equilibrium and/or balance the distribution of density in the system, though I will expect to see some oscillation as particles as the momentum fluctuates due to the wave like propagation of energy.

4.2 Test Results

4.2.1 Spring Approach Analytic

When I started I looked at what compression meant from a naive basic physical system, compression was when an existing mass was made to occupy a small volume of space. When a mass is placed into a small space it results in an increase of density in that volume and vice-versa pressure. This means that there should be retroactively some kind of force attempting to alleviate the pressure to some appropriate measure of compression of the mass.

I attempted to model this kind of behaviour through a physics spring constraint giving point masses a volume of influence such as a radial effect. The problem is adequate representation of the fluid volume is incorrect between an uncompressed state and a compressed state.

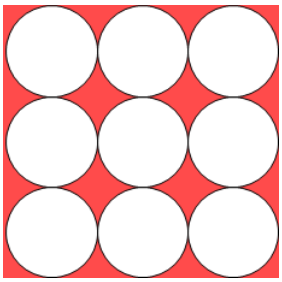

Figure 4: Depiction of Particle spread when packed into a space. The white region represents the volume of the particle, while the red regions would be the area of a volume with limited representation by the point particles.

External insight into the simulation would assume the fluid would represent and occupy that entire packed regions of particles, however under particle methods like SPH, in figure 4 regions marked in red would be inadequately represented internally within the fluid system, if it was to undergo compression so that the mass regions started to occupy and represent this space, the initial volume of a system would no longer be proportional for the system. In weak compressible methods or incompressible methods a large quantity of particles is used to better represent the fluid volume.

The other real problem with this method is it is entirely localised, it won't resolve compression with respect to the entire system, as it attempts to influence all surrounding particles entering its volume of influence.

4.2.2 Pressure Distribution Approach Analytic

The problem I have learnt with this approach is it is not suitable for propagating. The system could create a change of pressure such that it could cause some compressible behaviour to emerge however particles moving through the system don't propagate the energy as expected.

Only the addition of a viscosity term can be used to restrict the movement of particles through the fluid or pressure gradients in the opposing direction of the particle motion that it applies deceleration to the particles.

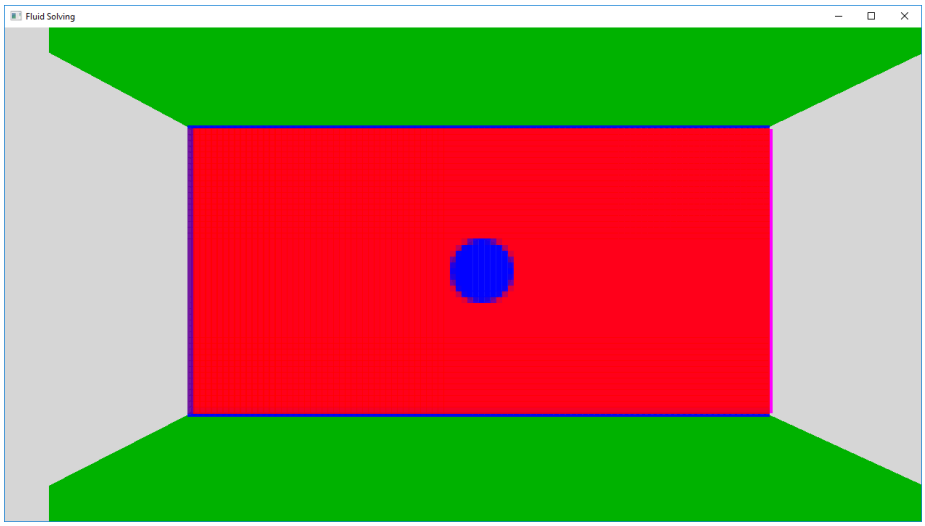

Figure 5: Depiction of initial pressure state for my test simulation. The heat map of my packed particles in a volume is a measure of my particle distribution in space. The blue region would identify as low pressure for the particles, and the red as high pressure.

It is important to note that the surrounding area's representing the solids (Green walls, and centred cylinder), may identify as a low pressure areas but handling of the different pressure regions is an important part of correctly handling the fluid behaviour.

4.2.3 Compressible Impulse Momentum Handling

All simulations use a numerical scalar value of 25 as a property of my fluid, it's used for dictating the compression restorative behaviour, basically how strong differences in pressure are at effecting the fluid.

The test generates an impulse in the centre of a enclosed box (figure 26) that has been filled with particles. After 10 seconds (600 iterations) an impulse force is released into the simulation.

The spring tests uses a different pattern for particle testing due to a difference in implemented behaviour and the performance cost of the simulation.

Spring Method 625 Particles

The box will have had the internal size readjusted to 25 units by 25 units with each particle having a diameter of 1. The impulse burst in the centre will also be rescaled to the change of volume add will apply the impulse in a small radius equal to half the box radius.

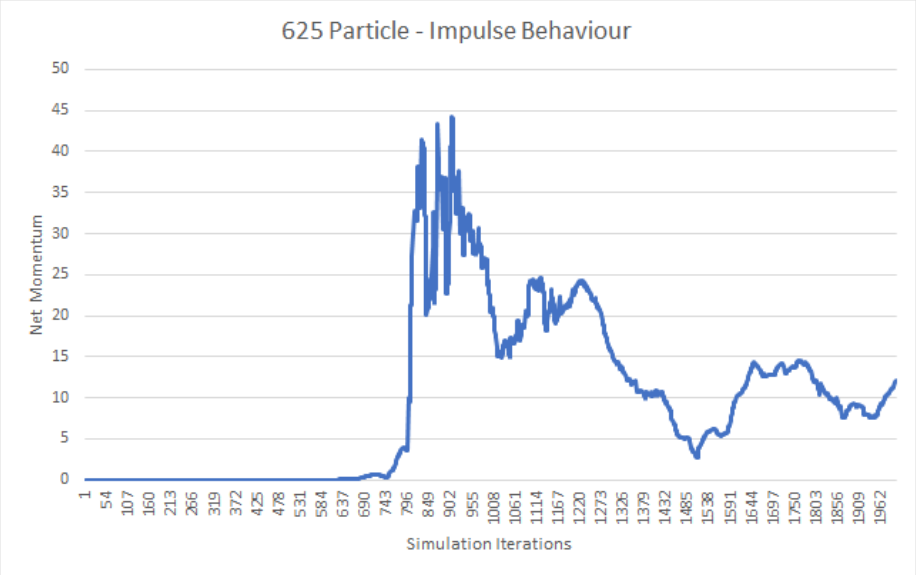

Figure 6: Momentum of Lagrangian Particle pressure method showing the momentum behaviour of the fluid over time.

Spring Method 2500 Particles

Looking at the peaks of the momentum in the graph, the simulation looks similar to when it is ran with less particles, except the spike in momentum for the system is more delayed at propagating the force of the impulse.

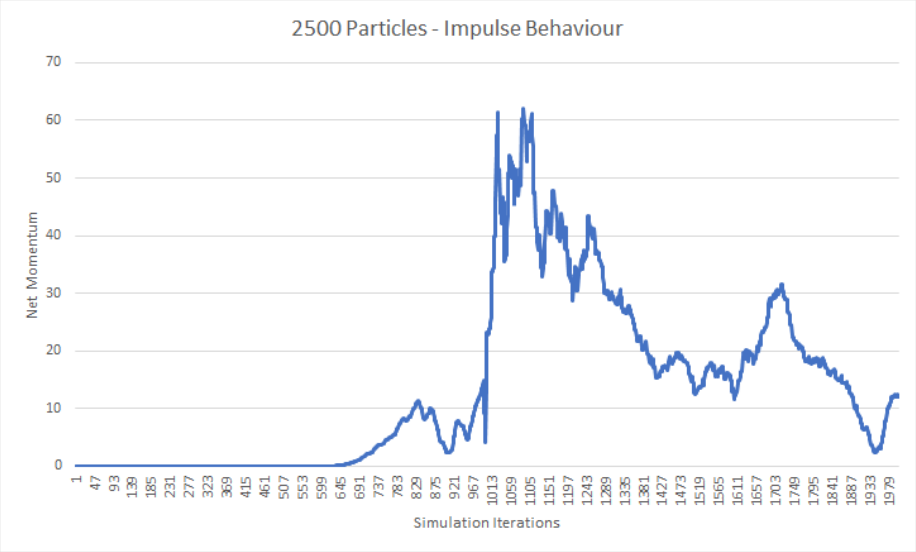

Figure 7: Momentum of Lagrangian Particle pressure method showing the momentum behaviour of the fluid over time.

Spring Method 10000 Particles

This is the highest number of particles that I am proposing to test initially the performance is not bad however after the impulse and velocity beings to propagate through the fluid system, the frame-rate quickly reaches the level of taking more than a second to complete a single iteration of the fluid solver.

Compared to the previous two simulations the behaviour is more out of sync, with the huge spike of energy that propagates the system happening much later into the simulation.

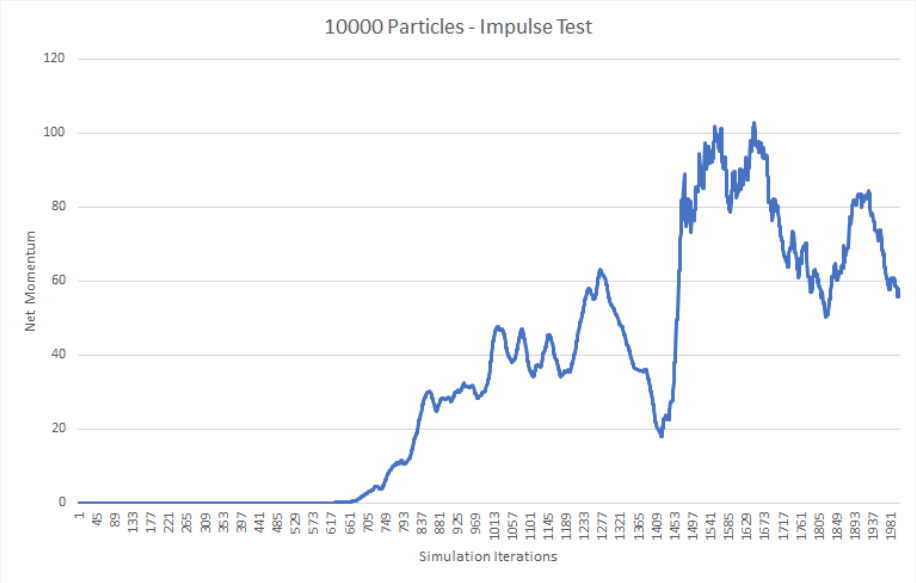

Figure 8: Momentum of Lagrangian Particle pressure method showing the momentum behaviour of the fluid over time.

Pressure Distribution Method 10000 particles

Test for how different densities effected the behaviour of the pressure resolution as seen by analysing the net momentum.

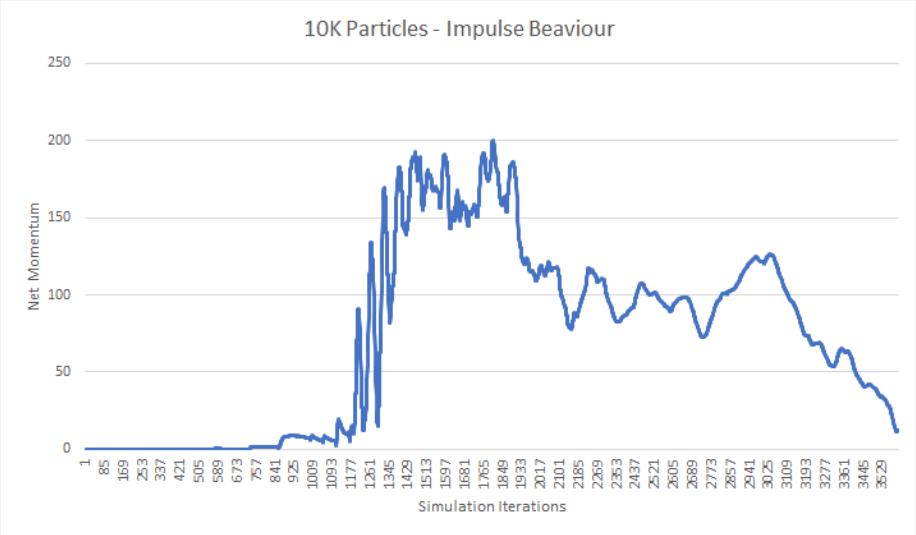

Figure 9: Momentum of Euler pressure method showing the momentum behaviour of the fluid over time.

Pressure Distribution Method 100000 particles

It's important to note that my framework limitation was negatively effecting this test, around the boundary cells as seen in figure 26 there is a very low region of density around the boundary cells, resulting in generation of flow towards the boundary. This effect is intensified at higher pressures which is why you can see a state of net momentum before the impulse is triggered.

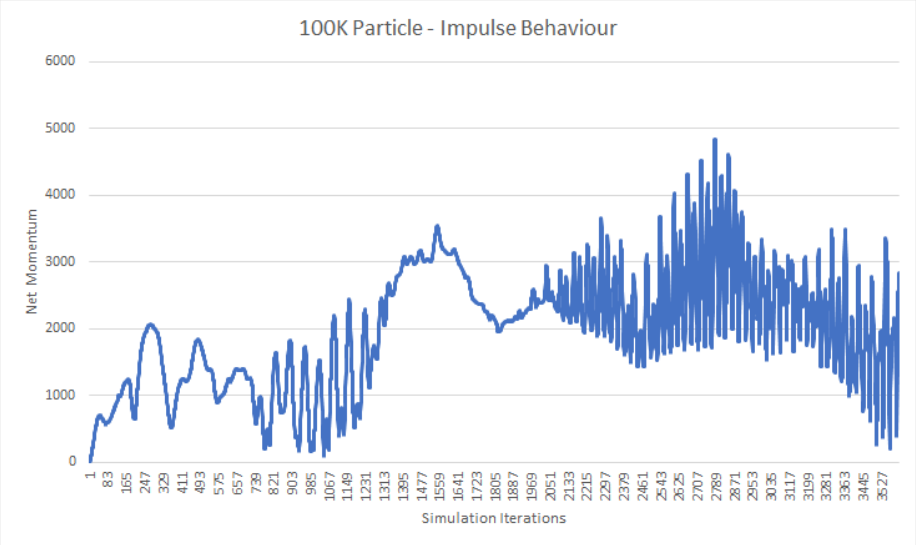

Figure 10: Momentum of Euler pressure method showing the momentum behaviour of the fluid over time.

Pressure Distribution Method 1000000 particles

Same as the previous test there is initial momentum generated from the high density fluid and the low pressure at the boundary.

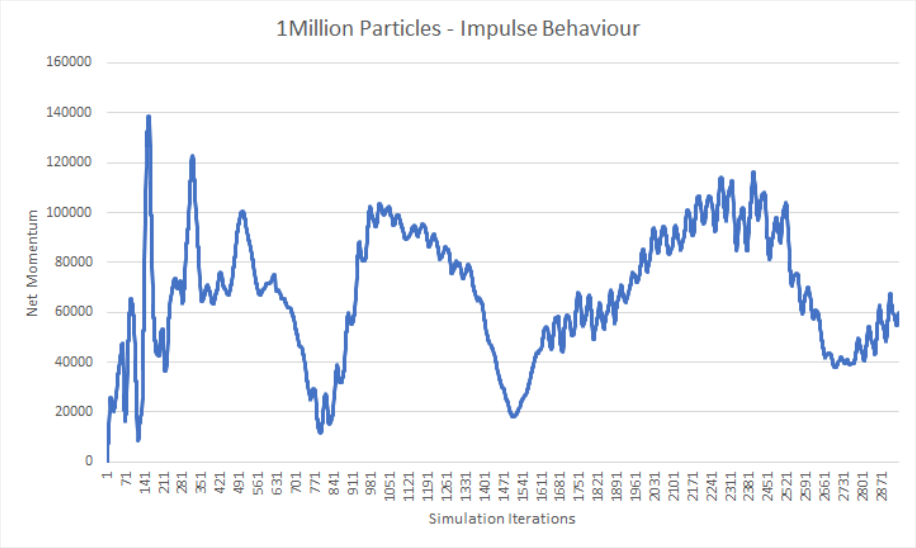

Figure 11: Momentum of Euler pressure method showing the momentum behaviour of the flow over time.

This simulation ran at 6 frames per second, limiting the potential to test the real-time behaviour at high particle counts.

4.2.4 Flow Behaviour of the Spring Method with Differing Viscosity - 500 Particles

I performed a flow test to try and understand how my Lagrangian spring method performed under varying viscosities.

Looking at the data in figure 15 and figure 13 I can see that the viscosity term causes similar behaviour on the flow of my fluid, with the velocity term effecting the gradient of the deceleration of fluid.

High Viscosity

A measure in fluid flow behaviour of the Lagrangian spring model with a viscosity value of 90%.

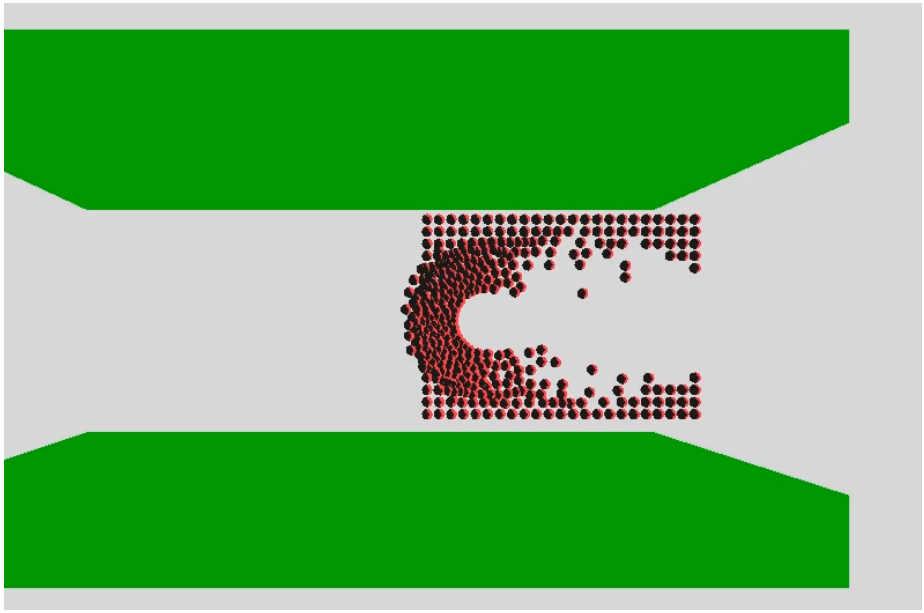

Figure 12: Image of the Lagrangian sprint method showing the momentum behaviour of the flow over time.

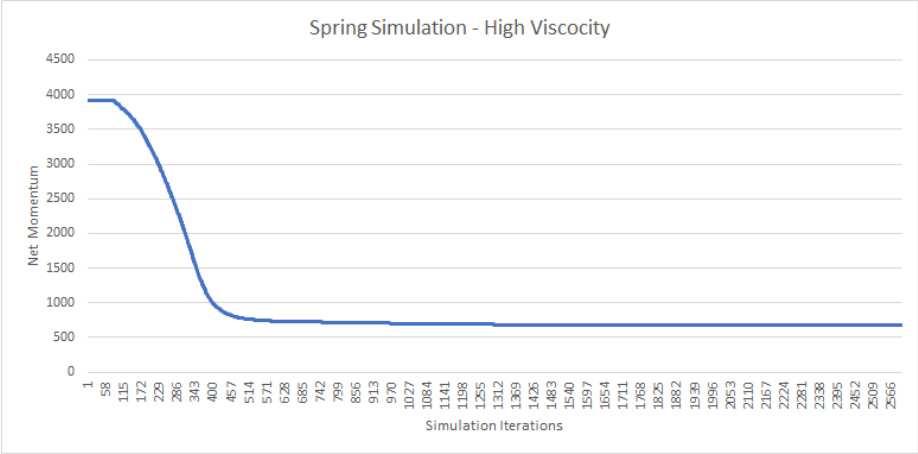

Figure 13: Image of the Lagrangian sprint method showing how the flow behaved with low viscosity.

Low Viscosity

A measure in fluid flow behaviour of the Lagrangian spring model with a viscosity value of 10%.

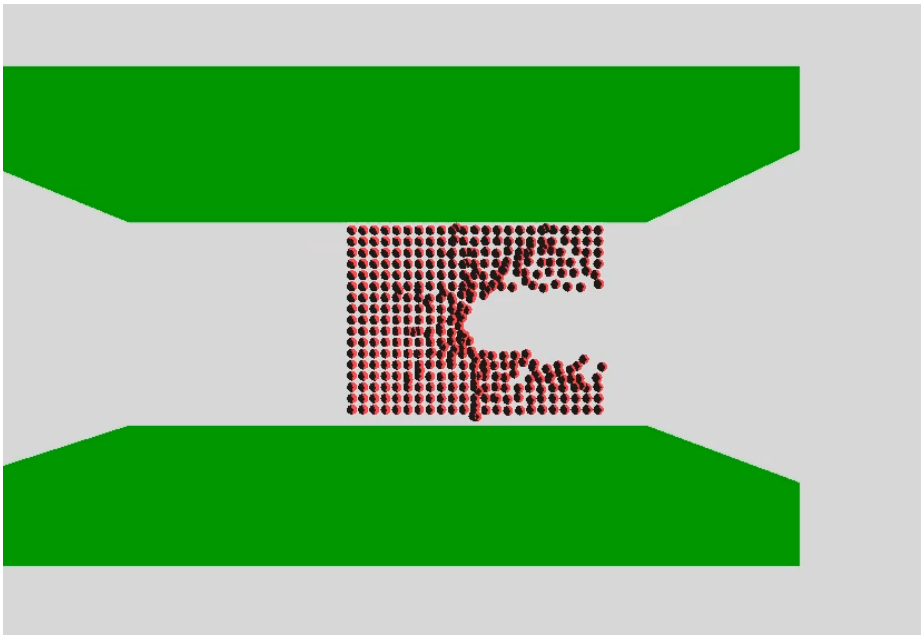

Figure 14: Image of the Lagrangian sprint method showing how the flow behaved with low viscosity.

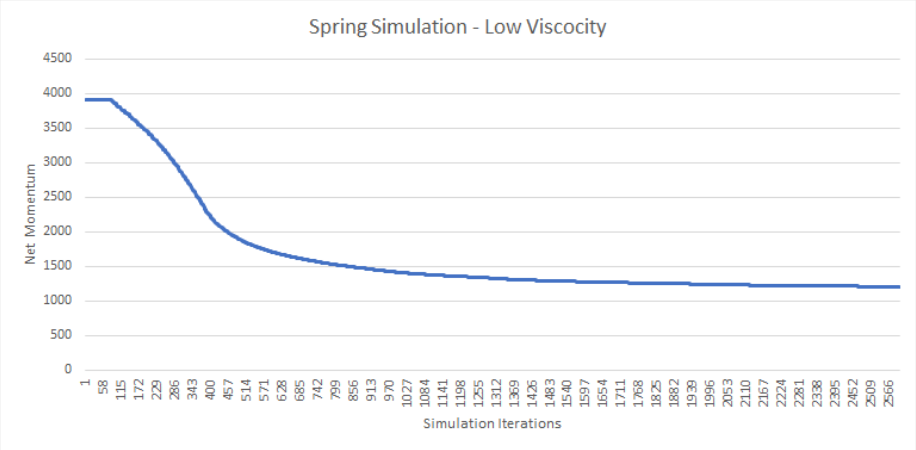

Figure 15: Image of the Lagrangian sprint method showing the momentum behaviour of the flow over time.

4.2.5 Compressible Flow Analysis of Pressure Grid Method

A created a fixed 100 unit long pipe that I moved my particle fluids through to see how they reacted to a blocking geometry. The idea was to try and evaluate the flow behaviour of my methods.

Behaviour No Boundary - 10000 Particles No Pressure Interpolation

This test was without using a boundary condition and instead directly using the collision engine to offset the particles to the boundary surface if they moved into the physical geometry.

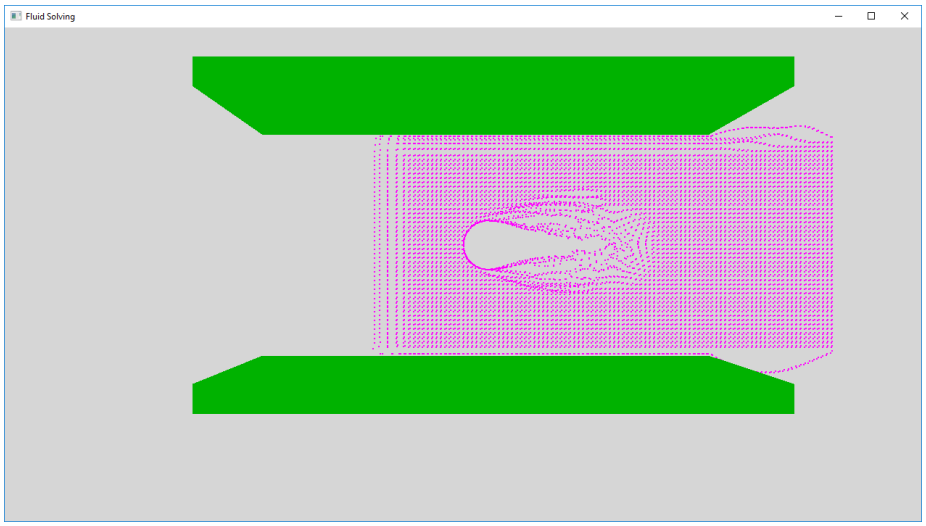

Figure 16: Image of the test showing how the flow is handled after collision with out a boundary condition without any interpolated pressure.

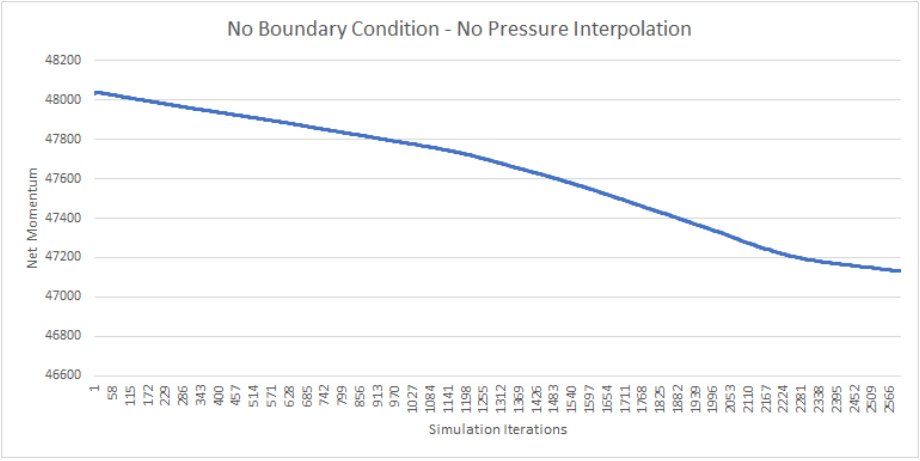

Figure 17: Net Momentum of test over time for my Euler pressure grid method without a boundary condition, without any interpolation of pressure.

Behaviour No Boundary - 10000 Particles Distance Pressure Interpolation

It is hard to see any visual differences between this interpolation and no interpolation method.

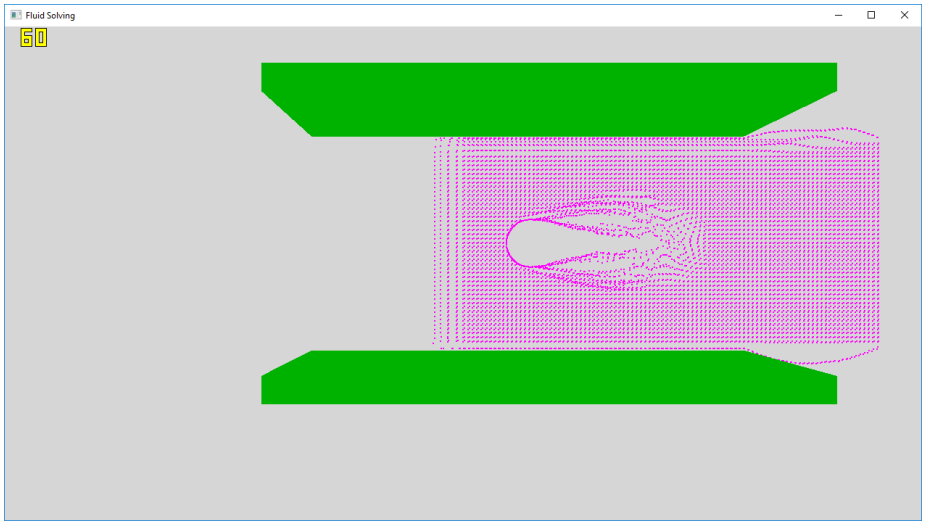

Figure 18: Image of the test showing how the flow is handled after collision with out a boundary condition using distance interpolated pressure.

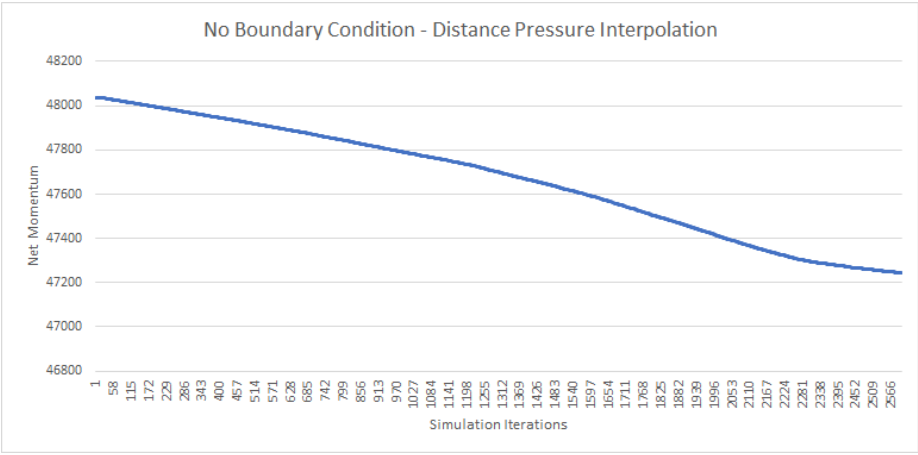

Figure 19: Net Momentum of test over time for my Euler pressure grid method without a boundary condition and using distance interpolation of pressure.

Behaviour No Boundary - 10000 Particles Bilinear Pressure interpolation

I noticed using bilinear interpolation caused the internal flow stream to be averaged out into a single direction, so a majority of the particles instead of being effected by the pressure change caused by the object instead get the restorative pressure nullified and continue on in a uniform path.

You can see in figure 20 how a majority of particles move with their initial velocity, with out being pushed into the low density space after the collision.

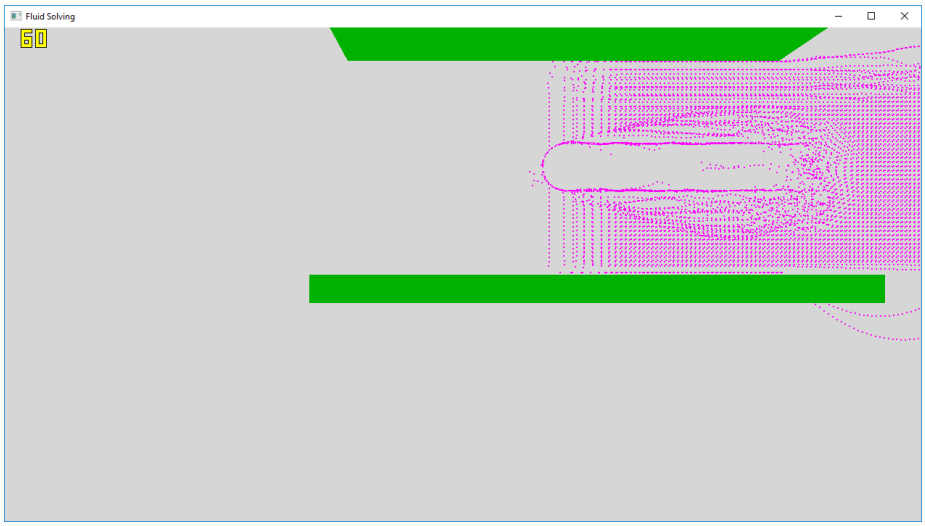

Figure 20: Image of the test showing how the flow is handled after collision with out a boundary condition using bilinear interpolated pressure.

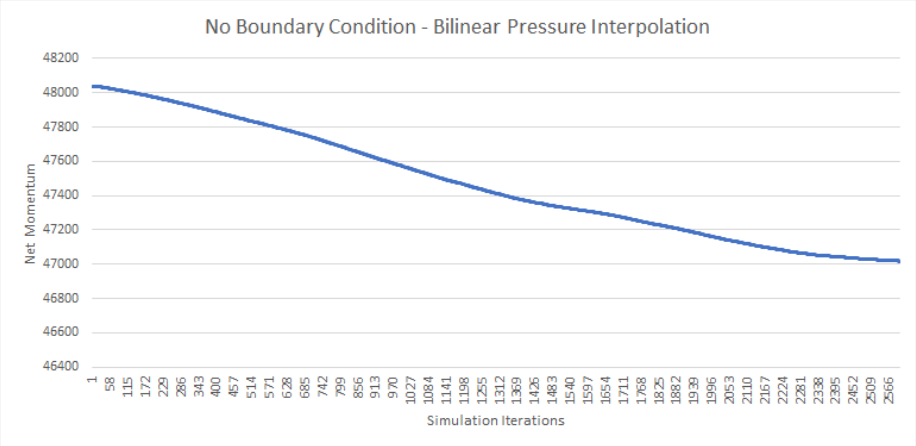

Figure 21: Net Momentum of test over time for my Euler pressure grid method without a boundary condition and using bilinear interpolation of pressure.

Behaviour Boundary - 10000 Particles No Pressure Interpolation

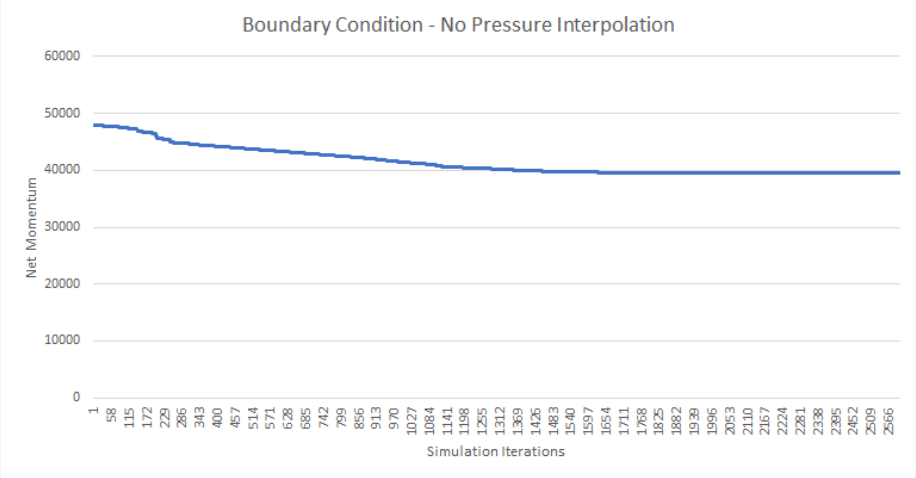

Figure 22: Net Momentum of test over time for my Euler pressure grid method using no interpolation of pressure.

Behaviour Boundary - 10000 Particles Distance Pressure Interpolation

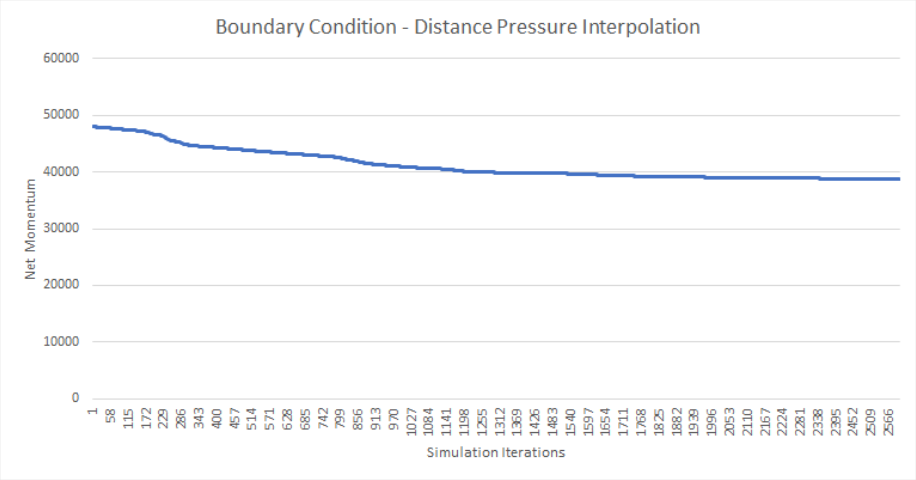

Figure 23: Net Momentum of test over time for my Euler pressure grid method using distance scaled interpolation of pressure.

Behaviour Boundary - 10000 Particles Bilinear Pressure interpolation

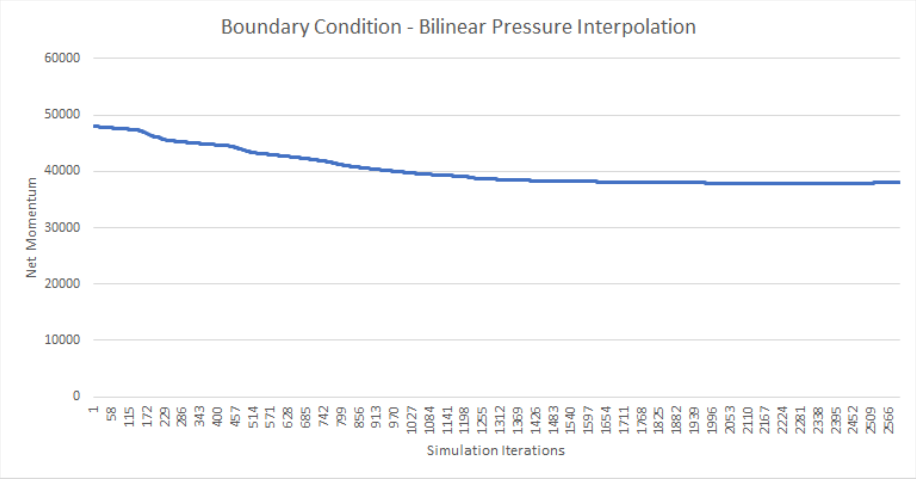

Figure 24: Net Momentum of test over time for my Euler pressure grid method using bilinear interpolation of pressure.

4.3 Analysis

Reminder to the previous limitation that some particles are removed from the simulation if spawned in a case where it can lead to numerical inaccuracies. This will be the case for particles in or close to physical boundary geometry.

4.3.1 Handling Impulse - Spring Separation

My spring method managed to correctly dampen into a lower momentum state after the initial burst of momentum that was injected into the system. However when I scaled the particles, in theory I expected similar behaviour between all the tests as the test scaling should have maintained a identical scale for the volume space to particle scale.

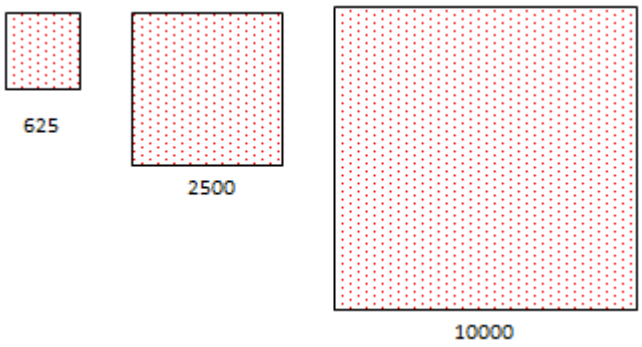

Figure 25: Example depiction of the scaling used in the impulse burst test on the spring like point particles, with consistent particle spread for the volume.

Though it was shown that at high particle counts like 10000 the behaviour changes, figure 8, shows that the impulse is slow to correctly propagate and did not experience a sharp peak of momentum like my other two tests. This means that my method is flawed, the propagation of force as a wave is not being correctly handled.

4.3.2 Handling Impulse - Pressure Grid

Looking at the generated data for the net momentum of the fluid at 10000 particles (figure 9) I can see that the initial impulse had very little effect on the net momentum initially, this would be because the movement of the particles are not being considered in this simulation method, as the particles move through the discretized spatial grid. The simulation will generate regions of high pressure and low pressure, which are then used to influence the flow of the fluid system. You do eventually see the momentum begin to balance out towards a zero velocity near the end of figure 9.

However it becomes tougher to evaluate when we scale up the mass in our volume (add more particles). The larger mass density means there is an initial flow towards the boundary, then once the impulse is injected into the system it is so compressed that it results in large fluctuations of the momentum as seen at the 1000th iteration of figure 10, while still gradually declining back into an initial momentum state.

4.3.3 Handling Flow - Viscosity Spring Approach

At low viscosity you can see the gradient for the decline (figure 15) in momentum of the system is less steep compared to the high viscosity momentum in figure 13.

The problem with the spring method is that the net momentum is not conserved, when the impact with the cylinder occurs there is a decline in the momentum which never gets restored. This means that the spring separation approach does not adequately handle flow.

4.3.4 Handling Flow - Pressure Grid

I tested different interpolation methods in conjunction with different handling for the boundary condition, I did this part to test that my boundary condition was acting correctly.

Handling Flow - Differences of Boundary Behaviour

Two near identical tests for the interpolation process was performed with the only difference being the boundary condition. One had my implementation of a no-slip boundary and the other had a simple penetration resolution boundary to prevent particles crossing into obstacles.

A comparison of net momentum of my penetration boundary and my no-slip boundary reveals that the penetration boundary failed to conserve the momentum of the system as it experienced a sharp drop in momentum (figures 17,19,21) compared to those from the no-slip boundary flow test (figures 22,23,24) which shows a small momentum drop.

Comparing these two I am under the belief that my boundary condition is behaving within minimal expectation of what it should be doing which is reducing the velocity of particles relative to that of the surface.

I believe this small momentum drop would be caused by particles that have become stuck at my boundary with zero velocity.

The results show within tolerable expectation that my boundary condition is performing correctly and that my pressure grid method is momentum conserving.

Handling Flow - How Differences In Pressure Interpolation Effected Flow

In the pressure system, the spatial cell centre is the origin for that cell it is where the pressure gradient the cell represents comes from. In terms of flow I didn't see any massive variations in behaviour except for the bilinear interpolation (figure 20), where you can see it being less effected by the low density regions after the cylinder, where as the other interpolation methods pushed the particles back into the centre looking somewhat like a laminar flow (figure 18, 16).

In terms of momentum in the system my interpolation has had minor effect on the momentum (other than the differences caused by the boundary condition), but a trend I can see is that the interpolated methods results in a lower final momentum in the order from no interpolation (figure 17, 22), to distance interpolated (figure 19,23) and finally bilinear interpolated (figure 21, 24).

4.4 Methodology Evaluation

4.4.1 Limitations

Testing Limitations

Compressible flow can be much harder to appropriately measure, the flow could end up generating turbulent flow which by nature is supposed to be chaotic and harder to predict, or it could generate a laminar flow. Traditionally chaotic flow would be measured with statistics however my system is very uniform, so some of the chaotic nature of flow can be lost so quantifying the data would hold a bias and be harder to correctly prove.

Testing the compressibility accuracy is a challenge and could be much harder to convert into a scalar, but I will be able to test my method for how it conserves momentum of the system under compression and while being restored to an equilibrium, I can measure the net momentum to see if compression is adequately handled by comparing my initial momentum to my final momentum.

Partly related to my issue with evaluating flow behaviour, my evaluation is also not very adept at evaluating curl and vorticity behaviour, as all I evaluate is the net energy in the system to measure energy behaviour.

Testing the flow behaviour under high pressure is hard to test. If I maintain a high pressure it is possible that my discretization is not smooth enough and can result in an uneven distribution of pressure that can result in the ejection of particles entering my flow chamber. I should have defined an inlet boundary condition of some sort to correctly handle injecting and ejecting particles in my flow simulation.

Testing the impulse behaviour for my pressure grid method has a small flaw, due to a framework limitation I have with processing dynamic objects that intersect the boundary generating invalid numerical states; I have to remove potential penetrating particles. The problem is my pressure grid system still interprets the boundary area as a region of low pressure, due to the removal of

particles within the boundary. This means initially it will result in the pressure test generating a flow towards the surrounding boundary.

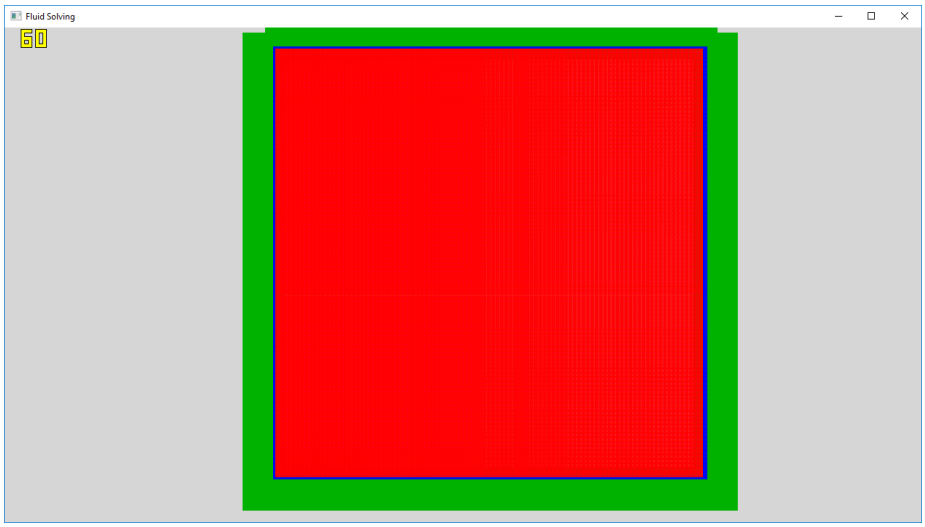

Figure 26: Initial Euler pressure grid for my particle system. Even though I have spawned the particles such that its aligned to an Euler pressure grid the boundary has been left as an empty region and has low pressure, which the pressure system will attempt to flow to.

Framework Limitations

My underlying simulation framework has some performance limitations which have impacted the experiments behaviour and potentially the results. These limitations are most notable in my rendering and collision detection processes.

When incorrectly handled these performance limitations could negatively impact the physics of my simulation. To reduce the potential problems caused by these technical limitations, I made my simulation run at a fixed update time cycle to enforce the physics to behave uniformly during each iteration of my simulation. That will provide me with a more stable uniform simulation however it does not remove the framework issues that have arisen.

Another step I took to optimise my collision detection process issues was to omit objects from collision detection checking if the object has zero velocity, therefore would have zero movement in space and would not have moved to collide with anything.

My spring based method for example is greatly effected by the collision detection framework limitation. For every particle added that is an additional object that needs to be processed in my collision detection process. Such that the worst case if all particles are moving has complexity of $O(N^2)$, which makes testing a large amount of particles with this method extremely expensive to test,

especially when testing with a magnitude of potentially thousands or millions of particles.

I made attempts to improve on the rendering process, especially because the performance impact in my system when trying to render over a million objects is a noticeable one. However making the process extremely efficient is not a simple task, there are many ways that this process could be improved but the scope required to properly optimise this process would take a lot more time than I had available. The performance optimization I was able to make was via using a mapped buffer implementation of my rendering fluid particles, so I could update existing rendering objects in the buffer rather than having to reload the data into a buffer.

Spawning dynamic objects such as particles intersecting with geometry can result in invalid numerical calculations for position and velocity of my particles in my simulated methods. This is due to my handling involving physical boundaries, not being able to handle being within the boundary (specifically being calculated on the incorrect side of the boundary) this exists for my rigid body handling system and my pressure boundary solution. To avoid this I have made it so my simulation cannot spawn particles initially intersecting geometry, as a result tests will be generated with a particle count in mind but the particle will be removed if it will cause numerical inaccuracies.

Method Limitations

Accuracy of my pressure system could have been better defined, my system worked by identifying a point mass and calculating what region it occupied my discrete spatial pressure grid. The limitation of this system is that even if I was to increase the grid resolution it limits where my mass can occupy. What I should be doing is based on position calculating a proportion of the mass representation into neighbouring cells.

My Pressure calculating grid system did not take into consideration particles laying on the border stride between other cells, as a result my density system does not effectively calculate a density to surrounding spatial regions.

The pressure system does not take into consideration the actual velocity of the particles, rather all it does is calculate a velocity to apply based on the pressure gradient.

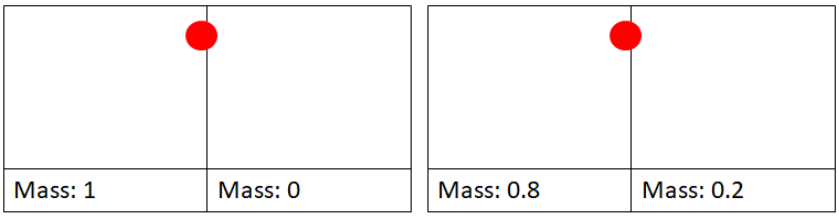

Figure 27: Presented is an arbitrary depiction of mass, on the left is how my system processes the mass on a point on the right is what my limitation should do, as this creates a smoother representation of my spatial discretization of mass, which increasing the grid resolution would not adequately remedy the discontinuity of mass.

One reason for avoiding this is the extra-processing cost is would cost to calculate each particle penetration into a cell. When used with a very large amount of particles it can be costly to calculate every frame if the particle rests on a boundary, and if it does calculate the division of mass into each cell. Potentially it could have been overcome by implementing a more precise and memory efficient method of handling bodies, their are many different types of dynamic spatial partitioning implementations out there, such as Octree's and AABB trees.

Moving the handling of the density away from the individual particles and into an observing system requires handling for processing the physical boundaries, my method of handling the boundary could be improved by either storing a reference to the geometry that makes the boundary and then performing relevant mathematics to the geometry to calculate where the surface is within that cell or storing an equation for the surface in the spatial cell.

4.5 Discussion

In my research and development I attempted two different methods of handling the fluid simulation with compressibility.

The first approach was with sphere particles using a Lagrangian based system. The compressibility was localized into the individual mass particle representation.

One thing I discovered about compressibility and for general fluid simulation is the complexity of calculating a discrete distribution of mass accurately that your system can act with appropriate levels of force. In my experiments I used a spherical particle to represent a unit of mass in the system, spheres would represent the mass volume, but this approach may not accurately represent the entire volume that fluid occupies (figure 4) . A large quantity of particles can be used to gain accuracy by decreasing the area that the particles are unable to represent, this is what is used in SPH to gain more accuracy.

My handling for calculating where mass was concentrated in a distribution

with my Lagrangian particle method was rather inefficient as it used collision detection to calculate overlapping particles to calculate mass intersection in the volume, to derived a density change. I should of done what other Lagrangian methods do which is get particles in range and then use them to calculate a density distribution while maintaining the separation logic separately like how other methods do the kernel smoothing.

Controlling the separation can be possible if my density distribution wasn't completely localised to the intersection. This method would attempt to just repel all particles regardless of the density distribution (or the derived pressure) around it.

My second approach was with a discretized grid volume using a Euler based system to calculate a pressure gradient to apply to on fluid particles residing in the spatial cell. Further usage of the pressure gradient was used to define a change in momentum on the particles. While functional there was issues related to expressing the change of momentum and controlling the effect applied to the fluid, a non-compressible fluid would apply a instantaneous change to the fluid or would have the grid dictate all the behaviour on particles.

Unlike the spring method the pressure grid method requires that external forces be calculated into the grid pressure; using a system of spatial defining pressure will need to utilize the previously mentioned pressure law for fluid under gravity $P = \rho G h$.

Handling the no-slip boundary, understanding the correct behaviour, should the velocity of anything at the boundary still take into consideration the pressure.

Considering the direct formula for pressure is Force divided by area it is interesting to investigate further if the boundary condition can be quantified as a more realistic force rather than a change of momentum based on the boundary.

Avoiding self propagation, it became apparent to me in testing that a single particle in an area can create flow in the surrounding volumes. Having large regions representing the pressure can cause incorrect handling of pressure, where the flow generated wouldn't correctly propagated and instead would generate

Chapter 5

Conclusions and Recommendations

Fluid simulations incorporate Fluid dynamic laws to calculate the behaviour of momentum of fluid systems. I experimented with smooth particle approaches of simulating fluid, and to incorporate compression I made my individual particles behave spring like incorporating surrounding point masses into the compressible spring behaviour. My attempted approach was simple to understand and set up by being based on existing real-time physics simulation knowledge. With the focus being on configuring the compressible behaviour of the fluid. However I have learnt that correctly adjusting this behaviour has proven to be difficult with out a firm understanding on how these properties should be integrated into a numerical solution for resolving compression.

In the spring particle approach the particles had an individually defined volume for the stable mass, with intersecting particles being able to repel based on a scalar interpretation of the compression, based on how the particle volumes overlapped, being used to calculate a percentage of mass existing within intersecting particle volumes, to calculate a change of density. I realised that this method has problems related to performance and how it doesn't effectively calculate a distribution of density.

Using a pressure grid system abstracted away the compressibility handling into an observing system which can be used to calculate a distribution of density in the fluid system. The pressure grid was used to resolve the distribution of density via changing the momentum of particles. This method using a discretized space, also has a few flaws related to processing the particle compression properties.

Comparing the pressure distribution method and the spring method. This grid method had the benefit of handling the fluid system collectively with proper respect for the mass occupation of volume, however it removes potential for self contained logical handling of particle interactions as the system transitions into an observing system that provides information on pressure.

My experimental approaches of handling compression worked along the basis of restorative behaviour, controlling how effective a fluid is able to return to a stable state via repulsive forces, such as ejecting a particle immediately that attempts to enter a high pressure region and scaling the effects of a high pressure region in the simulation. Applying restorative behaviour to the fluid also generates flow.

investigating the behaviour of the Lagrangian spring like particle approach, showed that the way I implemented it was far too computationally expensive to be used in real-time simulation, while also having instability at large particle counts. Though the computational cost can be debated down to the collision framework, my results show that it has inaccuracies at high mass representation.

I now know that calculating the fluid behaviour per-particle is more computationally expensive that having a system that can be used to query the fluid state, such as a discretized spatial grid for the density.

Numerical accuracies and relative unit scales were constant recurring issues for me initially, to the point where I would recommend using an already established framework or take care with the numerical precision.

5.1 Summary

In conclusion their are many different aspects to simulating fluids, with additional implementation complexity from compressibility with respect to the flow of the system.

Ultimately from my two approaches I developed, they were both adept at one thing but would be weak at another. The particle spring separation is good at controlling and defining the compressible behaviour but had issues with fluid flow behaviour and had higher computational costs. The pressure grid was computationally cheaper and was better at defining the particle flow but care has to be taken to effectively discretize your continuous system to get a good distribution of mass density. However the pressure grid method wasn't respective of existing velocity in the system or had a way of separating individual particles except at the discretized cell level.

Both of these issues I have encountered are things that are handled in existing simulation material. Euler grids can generate a velocity vector field and incorporate it into a calculation of flow (Euler or Navier-stokes), and a Lagrangian particle method like my spring approach would perform a query of surrounding particles within a specific range to generate a mass distribution to apply on the calculation for the particles flow.

Depending on what behaviour you want these methods could potentially be nice alternatives for simple compressible fluid behaviour. However if you want a compressible fluid simulation that is good at handling all properties of a compressible fluid, these methods will not be viable.

Chapter 6

Future Considerations

Having experimented with different methods of calculating a fluid density distribution I understand a bit more on the performance challenges involved with fluid simulation. Taking that into consideration it would be interesting to experiment more with my developed approach of the Lagrangian Spring Smooth Particle method.

Have the method take into consideration its own behaviour rather than just the mass compressional factor.

Future methods will require some form of collision query optimisation such as spatial hashing to more efficiently look up near by particles. I would also choose to work with a more reliable physics simulation framework rather than my own so I can better evaluate and develop my real-time simulations. Most physics engines are likely to have some form of collision query optimisation such as spacial hashing or spatial culling with Octree's.

Knowing how intensive my discretized pressure approach was, I would change my discretization method to follow other methods like the fluid simulations that utilize PDF. By using a more intuitive and less intensive approach for discretizing the density distribution for my point mass particles. A better discretization of density would allow for more accurate pressure calculations, enabling a more visually believable method and generate better data for the evaluation of my methods.

So far in this study I have focused exclusively on researching Navier-Stokes and Euler related research in fluid simulations. It is also worth investigating in future, Lattice Boltzmann as it is another form of Fluid dynamics, but to some degree it utilises collision which my particle method used to calculate a density.

Both methods I tried to today had they benefits and their negatives. It would be interesting to try and combine them and see how they behave. Using the spatial grid to optimize the particle collision detection query, using the grid to help calculate the pressure gradient and have the particle maintain the separation handling.

Bibliography

[sta, 2015] (2015). 1 edition.

[Amanifard and Namini, 2012] Amanifard, N. and Namini, V. H. (2012). A modified compressible smoothed particle hydrodynamics(mcsph) method and its application on the numerical simulation of low and high velocity impacts. *International Journal of Engineering*, 25(1):45–57.

[Becker and Teschner, 2007] Becker, M. and Teschner, M. (2007). Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217. Eurographics Association.

[Bindel, 2011] Bindel, D. S. (2011). Applications of parallel computers (cs 5220).

[Bridson and Müller-Fischer, 2007] Bridson, R. and Müller-Fischer, M. (2007). Fluid simulation: Siggraph 2007 course notes video files associated with this course are available from the citation page. In *ACM SIGGRAPH 2007 courses*, pages 1–81. ACM.

[Coumans et al., 2013] Coumans, E. et al. (2013). Bullet physics library. *Open source: bulletphysics. org*, 15.

[Cui, 2009] Cui, C. C. (2009). Fluid simulation overview.

[Gregory, 2009] Gregory, J. (2009). *Game engine architecture*. CRC Press.

[Lind et al., 2015] Lind, S., Stansby, P., Rogers, B., and Lloyd, P. (2015). Numerical predictions of water–air wave slam using incompressible–compressible smoothed particle hydrodynamics. *Applied Ocean Research*, 49:57–71.

[Müller-Fischer,] Müller-Fischer, M. Real time fluids in games.

[Pope, 1994] Pope, S. (1994). Lagrangian pdf methods for turbulent flows. *Annual review of fluid mechanics*, 26(1):23–63.

[Price, 2011] Price, D. J. (2011). Smoothed particle hydrodynamics: things i wish my mother taught me. *arXiv preprint arXiv:1111.1259*.

[Stam, 1999] Stam, J. (1999). Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co.

[Stam, 2003] Stam, J. (2003). Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, volume 18, page 25.

[Welton and Pope, 1997] Welton, W. C. and Pope, S. B. (1997). Pdf model calculations of compressible turbulent flows using smoothed particle hydrodynamics. *Journal of Computational Physics*, 134(1):150–168.

[West, 2008] West, M. (2008). Gamasutra - practical fluid dynamics: Part 1.

6.1 Appendices