Investigation into Smoothed Particle Hydrodynamics for Non-Newtonian Droplet Modelling

by

Gavin Lobo

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Faculty of Graduate Studies (Modelling and Computational Science)
University of Ontario Institute of Technology

Supervisor(s): Dr. Faisal Qureshi and Dr. Dhavide Aruliah

Abstract

Investigation into Smoothed Particle Hydrodynamics for Non-Newtonian Droplet Modelling

Gavin Lobo

Master of Science

Faculty of Graduate Studies

University of Ontario Institute of Technology

2011

Droplet splatter dynamics is an important study in the field of forensics since a crime event can produce many blood stains. Understanding the origins of the blood stains from pure observations is very difficult because much of the information about the impact is lost. A theoretical model is therefore needed to better understand the dynamics of droplet impact and splatter. We chose to explore a fluid modelling method known as Smoothed Particle Hydrodynamics (SPH) to determine whether it is capable of modelling droplet splatter accurately. Specifically, we chose to investigate an SPH version of a non-Newtonian pressure correction method with surface tension. Three experiments were performed to analyze the different aspects of SPH. From the results of the experiments, we concluded that this method can produce stable simulations if an artificial viscosity model is included, a third-order polynomial kernel is used and the pressure boundary condition on surface particles are non-zero.

Dedication

To my parents and my brother, Calvin.

Contents

Chapter 1

Introduction

1.1 Objective

Bloodstain pattern analysis is the science of looking at the stain blood droplets make on surfaces and attempt to determine the event that caused the stain. In many cases, it involves looking at the bloodstain and reconstructing the trajectories of individual blood droplets to determine their individual points of origin. This method, known as backtracking, works by analyzing the stain's dimensions and using some basic trigonometric relations, the direction of impact of the blood can be determined [9]. Once the direction of the impact is determined, rays are projected backwards along the angle of impact. The intersection of all the rays from different bloodstains determine the approximate location of the origin of the blood. This method can only be used for very specific types of stains, such as elliptical stains. Not all blood droplets produce elliptical stains so using the aforementioned method cannot be used. Understanding the origins of the blood stains from pure observations is difficult because much of the information of the impact is lost (such as velocity of impact). A theoretical model is therefore needed to better understand the dynamics of droplet impacts. The objective of this thesis is to work towards developing a model to simulate non-

Newtonian droplets in hopes that it will be used to better understand how various blood stains are produced.

To investigate the dynamics of blood droplets, we chose a method known as Smoothed Particle Hydrodynamics (SPH) as a simulation technique. SPH was chosen for a number of reasons; firstly, not much is known about the full capabilities of SPH so we chose to investigate this method further to determine its ability to simulate fluid motion; second, it is easier to implement compared to other methods; finally, SPH is quite customizable. Adding new physical effects to the fluids such as heat transfer or magnetic fields, for example, is relatively easy compared to the other methods of simulation. The objective of this thesis is to investigate the capabilites of SPH to determine whether it is a suitable modelling technique to use to understand the dynamics of blood droplets.

1.2 Related Work

There are many methods available for simulating fluids, some of which are finite difference methods and finite element methods [15]. Each of these have their advantages and disadvantages and are better suited to solve different types of problems. Finite difference methods, solve the solution on very structured, rectangular, grids and are best used on domains where the boundary walls are at 90 degree angles from each other. Finite element methods, on the other had, solve the solution on unstructured grids. This allows the boundaries of the domain to be non-rectangular. Both finite difference and element methods are known as fixed-grid methods. Like the name implies, the grid (structured or unstructured) the solution is solved on does not move, it remains fixed for the entire duration of the simulation. SPH is known as a mesh-free method. In mesh-free methods, the solution is not bound to a fixed mesh. The node points at which the solution is solved can move in time. In SPH, the nodes that the solution is solved at act like particles of fluid that move through the domain. Essentially, SPH is a cross between a particle method and an interpolation method.

SPH was originally developed in the 1970s to model astrophysical phenomena such as large scale gas dynamics in galaxies[5] [11]. Since then, there have been many advancements in the technique which have allowed it to simulate not only compressible fluids at the astrophysical scales, but also incompressible fluids at more commonplace terrestrial length scales. In traditional SPH, the mass density of the fluid is determined by the number density of particles within a certain volume of space (See Section 2.3).The density is then used to calculate other properties of the fluid such as the pressure [12]. This method, known as the *summation density* [13] approach, works well for modeling gas dynamics; gases are highly compressible, and, moreover, the summation density approach is guaranteed to conserve mass. For fluids of lower compressibility, the summation density approach is not physically reasonable. Instead, the equation of continuity is evolved in time to compute the density [13]. The density is again used to calculate the pressure from some kind of equation of state. Both, summation density approach and continuity equation approach assume that the fluid is compressible, which does not work well with fluids that are considered incompressible unless the equation of state is very stiff, i.e., small changes in density lead to large changes in pressure. A stiff equation of state requires much smaller step sizes in the time-integration, which is not very desirable since the run time of the simulation would be very long. A projection method was developed by Cummins and Rudman that enforces incompressibility by using an SPH derived pressure correction method [3]. This method allows for larger time steps to be used and guarantees the flow is incompressible by staying divergence free.

SPH can be used to model fluids at many different scales. For fluids with free surfaces at large scales, such as lakes or rivers, surface tension does not need to be explicitly modeled into the equations since the length scales at which surface tension forces apply are small compared to the spatial resolution. But on small length scales, a correct model of surface tension must be implemented because it highly influences the flow of the fluid. To model surface tension in SPH, particles that appear on the surface must be differentiated from particles that appear inside the fluid. The most common method of identifying surface particles is to see which particles have a density lower than the reference density. Particles near the surface will have a lower density due to there not being enough particles in the neighbourhood to contribute to the density calculation. Surface normals and curvatures can be found by computing the first and second derivatives of the density field [14]. Tartakovsky and Meakin modeled surface tension by using a pairwise force that attracted nearby particles similar to a Lennard-Jones potential[18]. Zhang et al used the actual Lennard-Jones potential to model surface tension [20]. Haque and Dilts used a different method for identifying surface particles. Rather than using the density of the particles to determine which particles appear on the surface, surface particles are determined by constructing imaginary circles (spheres in three dimensions) around each particle and determining if a particle's circle is completely overlapped by its neighbours circles [6]. A particle is identified as a surface particle if its circle is not fully covered by neighbouring circles. Once surface particles are found, the surface curve is locally constructed at each surface particle using a moving least squares method. The surface normals and curvature can be calculated analytically from the reconstructed surface curve.

Chapter 2

Introduction to Smoothed Particle Hydrodynamics

Smoothed particle hydrodynamics is an interpolation method used to discretize partial differential equations over a Lagrangian grid. Unlike Eulerian grids, where the grid nodes do not move with time, Lagrangian grid nodes move according to some flow field. In the case of fluids, the grid nodes move with the velocity field. The grid nodes in SPH also represent particles of fluid that contain mass. These particles are not fluid molecules, but represent some volume of the fluid depending on the scale of the system.

2.1 Formulation of Smoothed Particle Hydrodynamics

2.1.1 Function Approximation

The formulation of smoothed particle hydrodynamics starts with the identity for a scalar function

$$f(\mathbf{r}) = \int_{\Omega} f\left(\mathbf{x}\right) \delta\left(\mathbf{r} - \mathbf{x}\right) d\mathbf{x}, \tag{2.1}$$

where $f(\mathbf{r})$ is a scalar function of a vector variable, Ω is the entire spatial domain and $\delta(\mathbf{r}-\mathbf{x})$ is the Dirac delta function that is commonly seen in the SPH literature as

$$\delta\left(\mathbf{r}-\mathbf{x}\right)=\begin{cases}+\infty, & \mathbf{r}=\mathbf{x}\\ 0, & \mathbf{r}\neq\mathbf{x}\end{cases} \tag{2.2}$$

In Equation 2.1, the delta function will be zero for all $\mathbf{r}\neq\mathbf{x}$, so the integration will yield the value of the function where $\mathbf{x}=\mathbf{r}$. In numerical integration, the delta function is impossible to use in its current form due to the fact that its value goes to infinity, so it must be approximated with another function. In SPH, the delta function is approximated with a Gaussian-like function called a kernel or smoothing function. The exact form of the smoothing function can vary, but must have the following properties:

$$\int_{\Omega} W(\mathbf{r},h)dV=1 \qquad \text{(the Normalization Condition);} \tag{2.3}$$

$$\lim_{h\to 0} W(\mathbf{r},h)=\delta(\mathbf{r}) \qquad \text{(the Delta Function Property); and} \tag{2.4}$$

$$W(\mathbf{r},h)=0, |\mathbf{r}|>h \qquad \text{(Compact support)} \tag{2.5}$$

In Equation 2.4 the parameter h is a scalar constant known as the smoothing length, it represents how wide the smoothing function is. In addition to the above properties, the smoothing function should also be positive, even and monotonically decreasing away from the origin.

When the delta function δ is replaced by a particular smoothing kernel W in the integrand of Equation 2.1, we obtain

$$f(\mathbf{r})\approx\int_{\Omega} f\left(\mathbf{x}\right)W\left(\mathbf{r}-\mathbf{x},h\right)d\mathbf{x} \tag{2.6}$$

as an integral representation of a function $f(\mathbf{x})$.

When approximating a function using the kernel, the following convention is adopted in the SPH literature rather than using the $\approx$ symbol [10].

$$\langle f(\mathbf{r})\rangle = \int_{\Omega} f(\mathbf{x})\, W(\mathbf{r}-\mathbf{x},h)\, d\mathbf{x} \tag{2.7}$$

Consider the one-dimensional case of Equation 2.7. By Taylor Expanding the function $f(x)$ around the point $f(r)$ we get

$$\begin{aligned}
\langle f(r)\rangle &= \int_{\Omega} \left[f(r) + f'(r)(x-r) + O(|x-r|^2)\right] W(r-x,h)dx && (2.8)\\
&= f(r)\int_{\Omega} W(r-x,h)dx + f'(r)\int_{\Omega}(x-r)W(r-x,h)dx \\
&\quad + \int_{\Omega} O(|x-r|^2)W(r-x,h)dx && (2.9)\\
&= f(r)\int_{\Omega} W(r-x,h)dx + f'(r)\int_{\Omega}(x-r)W(r-x,h)dx \\
&\quad +O(h^2), && (2.10)
\end{aligned}$$

where O is the limiting behaviour of the Taylor Expanded function that is truncated at its linear term. In Equation 2.10, the integral in the first term is equal to 1 by the normalization condition (Equation 2.3). The integrand in the second term is a product of an odd and an even function (assuming W is chosen to be even), therefore the integration over the domain will reduce to zero and the function approximation reduces to

$$\langle f(r)\rangle = f(r) + O(h^2) \tag{2.11}$$

As can be seen from the above equation, the integral approximation using the smoothing function, in replacement of the delta function, yields an error that is proportional to h^2. This derivation assumes that the smoothing function is symmetric about it's origin. Non-symmetric smoothing functions do exist, but they are only used on the boundaries of the domain [10].

2.1.2 Derivative Approximation

To obtain the derivative of a function, we can replace the function value, $f(\mathbf{r})$, in equation 2.7 with the gradient of $f(\mathbf{r})$.

$$\langle \nabla f(\mathbf{r}) \rangle = \int_{\Omega} [\nabla f(\mathbf{x})] W(\mathbf{r}-\mathbf{x},h) d\mathbf{x}, \tag{2.12}$$

By using a rearrangement of the product rule

$$[\nabla_{\mathbf{x}} f(\mathbf{x})] W(\mathbf{r}-\mathbf{x}) = \nabla_{\mathbf{x}} [f(\mathbf{x}) W(\mathbf{r}-\mathbf{x},h)] - f(\mathbf{x}) \nabla_{\mathbf{x}} W(\mathbf{r}-\mathbf{x},h) \tag{2.13}$$

and substituting it into equation 2.12, we are left with

$$\begin{aligned} \langle \nabla_{\mathbf{r}} f(\mathbf{r}) \rangle &= \int_{\Omega} \nabla_{\mathbf{x}} [f(\mathbf{x}) W(\mathbf{r}-\mathbf{x},h)] \, d\mathbf{x} - \int_{\Omega} f(\mathbf{x}) \nabla_{\mathbf{x}} W(\mathbf{r}-\mathbf{x},h) d\mathbf{x} & (2.14) \\ &= \int_{S} f(\mathbf{x}) W(\mathbf{r}-\mathbf{x},h) \mathbf{n} dS - \int_{\Omega} f(\mathbf{x}) \nabla_{\mathbf{x}} W(\mathbf{r}-\mathbf{x},h) d\mathbf{x} & (2.15) \end{aligned}$$

The first term in the second step is replaced with a surface integral using the divergence theorem. Since the smoothing function goes to zero at a finite distance away from $\mathbf{x}$, the surface integral is therefore equal to zero and we are left with the integral approximation of the derivative of the function f.

$$\langle \nabla_{\mathbf{r}} f(\mathbf{r}) \rangle = -\int_{\Omega} f(\mathbf{x}) \nabla_{\mathbf{x}} W(\mathbf{r}-\mathbf{x},h) d\mathbf{x} \tag{2.16}$$

Using the integral representation of the function shifts the derivative from the function onto the kernel. The kernel, since it is a known analytic function, can easily be differentiated. Equation 2.17 can be used to compute the gradient of a scalar function. By following the same derivation, for a vector function, we are left with a similar equation to compute its divergence,

$$\langle \nabla_{\mathbf{r}} \cdot \mathbf{f}(\mathbf{r}) \rangle = -\int_{\Omega} \mathbf{f}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} W(\mathbf{r}-\mathbf{x},h) d\mathbf{x} \tag{2.17}$$

The accuracy of the function approximations is $O(h^2)$ but the accuracy of the derivatives is close to $O(h)$. Refer to section 4.2 for additional information.

2.2 Kernel Functions

The kernel function, W, can be chosen from many possible classes of functions. Qualitatively, the graph of a one-dimensional kernel function typically resembles a radially-symmetric Gaussian bell-shaped curve. However, in accordance with Equation 2.5, any kernel function necessarily has compact support.[1] The kernels that we will be using are the sixth order polynomial kernel, $W_6(r)$, and the third order polynomial kernel, $W_3(r)$. The kernels are usually defined as a function of radial distance between two particles. The shapes of these kernels are shown in Figures 2.1 and 2.2.

$$W_6(r,h) = \begin{cases} A_6\left(1-\left(\frac{r}{h}\right)^2\right)^3 & ,r<h \\ 0 & ,\text{Otherwise} \end{cases} \tag{2.18}$$

$$W_3(r,h) = \begin{cases} A_3\left(1-\left(\frac{r}{h}\right)\right)^3 & ,r<h \\ 0 & ,\text{Otherwise} \end{cases} \tag{2.19}$$

The constants A_6 and A_3 are the normalizing constants required to make the spatial integral over the kernel's domain equal to 1. The normalizing constants for one, two and three dimensions are defined in the following table.

[1]Recall that a function $W : \mathbb{R}^n \to \mathbb{R}$ is said to have *compact support* if W is everywhere continuous and there exists $h > 0$ such that $W(\mathbf{x}) = 0$ whenever $|\mathbf{x}| \geq h$

Coefficient	1D	2D	3D
A_6	$\frac{35}{32\pi h}$	$\frac{4}{\pi h^2}$	$\frac{315}{64\pi h^3}$
A_3	$\frac{2}{h}$	$\frac{10}{\pi h^2}$	$\frac{15}{\pi h^3}$

Table 2.1: Coefficients for the different kernels in one, two and three dimensions.

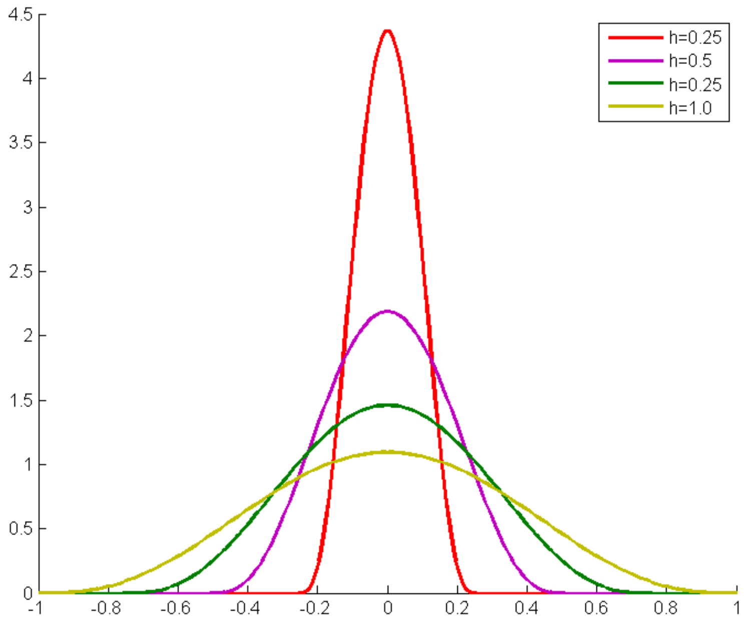


Figure 2.1: Sixth-order Guassian-like kernel.

2.3 Particle Approximation

In SPH, the domain is filled by a number of particles that hold various local properties of the system, for example: temperature, energy, mass, etc. These particles take up a finite volume in the domain. The particle approximation occurs when the integral in equation 2.7 is replaced by a summation.

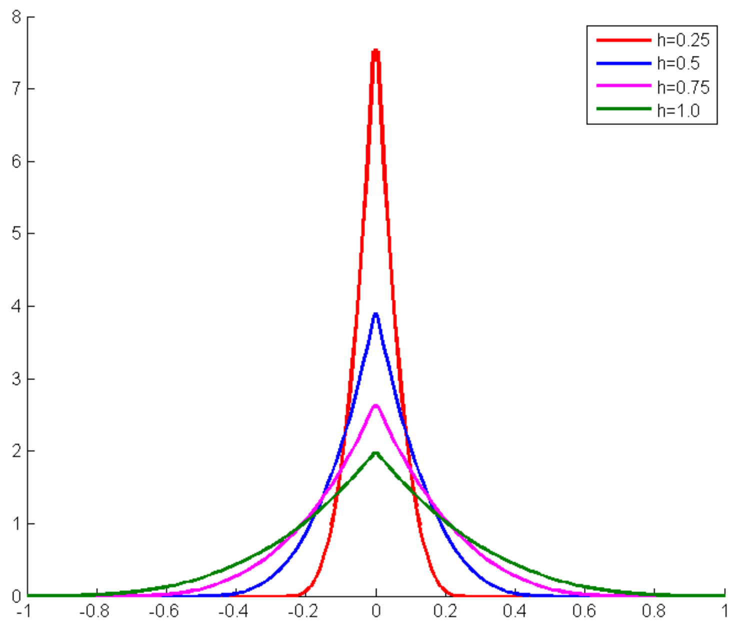


Figure 2.2: Third-order polynomial kernel, also known as the Spikey kernel

$$\langle f(\mathbf{r})\rangle = \int_\Omega f(\mathbf{x})\, W(\mathbf{r}-\mathbf{x},h)\, d\mathbf{x} \tag{2.20}$$

$$= \sum_{j=1}^{N} f_j W\left(\mathbf{r}-\mathbf{r}_j,h\right) V_j \tag{2.21}$$

$$= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W\left(\mathbf{r}-\mathbf{r_j},h\right) \tag{2.22}$$

Here, $\mathbf{r}$ is any position in space, $\mathbf{r}_j$ is the position of particle j, f_j is the value of the function that particle j holds. For example the temperature at the particle's location. The spatial volume, V_j is replaced with the mass, m_j of particle j divided by its density, ρ_j. The summation is taken over all particles in the system, but since the smoothing function goes to zero after a certain distance away from $\mathbf{r}$, only particles within that distance need to be included in the summation, the rest can be ignored.

In many cases, we are concerned with a quantity or a derivative at a particle's location rather than at some arbitrary location in space. The following convention is used when computing the particle approximation at the location of particle i:

$$\begin{aligned}
\langle f \rangle_i &= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W\left(\mathbf{r_i} - \mathbf{r_j}, h\right) && (2.23) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W\left(|\mathbf{r}_{ij}|\right) && (2.24) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W_{ij} && (2.25) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W_{ij} && (2.26)
\end{aligned}$$

where the following notation is used: $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, and $W_{ij} = W(\mathbf{r}_{ij})$. Note here that the smoothing function, W, is usually a radially symmetric function and is most typically defined as a function of distance rather than a function of position. Likewise, the derivative of f in the particle approximation can be obtained by

$$\begin{aligned}
\langle \nabla f \rangle_i &= -\sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} \nabla_j W\left(\mathbf{r_i} - \mathbf{r_j}, h\right) && (2.27) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} \nabla_i W\left(\mathbf{r_i} - \mathbf{r_j}, h\right) && (2.28) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} \nabla_i W\left(|\mathbf{r}_{ij}|\right) && (2.29) \\
&= \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} \nabla_i W_{ij} && (2.30)
\end{aligned}$$

where ∇_i the gradient with respect to particle i's coordinates and ∇_j is the gradient with respect to particle j's coordinates. In Equation 2.28, the gradient operator has switched to particle i's coordinates thereby removing the negative from the equation. Since the smoothing function, W, is a function of radial distance, $|r|$, the gradient of the smoothing function must be determined from the chain rule.

$$\begin{aligned}
\nabla_i W_{ij} &= \frac{\partial W}{\partial r}\nabla_i r && (2.31)\\
&= \frac{\partial W}{\partial r}\nabla_i \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right) && (2.32)\\
&= \frac{\partial W}{\partial r}\frac{1}{|r_{ij}|}\mathbf{r}_{ij} && (2.33)\\
&= \left(\frac{\partial W(|\mathbf{r}_{ij}|)}{\partial r}\cdot\frac{1}{|\mathbf{r}_{ij}|}\right)\mathbf{r}_{ij} && (2.34)
\end{aligned}$$

It is important to note that the term within the bracket in Equation 2.34 can be non-singular at $|r_{ij}| = 0$ depending on which kernel is used. The sixth order polynomial is non-singular at $|r_{ij}| = 0$ while the third order polynomial is singular.[2]

The density of each particle, ρ_i, is determined using the particle approximation of the density itself:

$$\begin{aligned}
\langle\rho\rangle_i &= \sum_{j=1}^{N} \rho_j \frac{m_j}{\rho_j} W\left(\mathbf{r}_i - \mathbf{r}_j, h\right) && (2.35)\\
\langle\rho\rangle_i &= \sum_{j=1}^{N} m_j W\left(\mathbf{r}_i - \mathbf{r}_j, h\right) && (2.36)
\end{aligned}$$

Here, the subscript i on the left side of Equation 2.36 indicates the particle approximation to the quantity within the angled brackets at the location of particle i.

The density of each particle can be calculated using Equation 2.36, after which any property (temperature for instance) can be calculated in any location in space using Equation 2.22.

2.4 Alternative methods for computing gradients

Consider the following identity obtained from the product rule of derivatives.

[2]When implementing this in code, it is better to compute $\frac{\partial W_{ij}}{\partial r}\frac{1}{|r_{ij}|}$ as one value rather than computing $\frac{\partial W_{ij}}{\partial r}$ and $\frac{1}{|r_{ij}|}$ by themselves and then multiplying the two terms together.

$$\nabla f(\mathbf{x}) = \frac{1}{\rho}\left(\nabla(\rho f(\mathbf{x})) - f(\mathbf{x})\nabla\rho\right) \tag{2.37}$$

By applying the derivative approximation (Equation 2.30) to all the gradient operators and leaving the particle values for all variables that are not part of the gradient, we obtain

$$\begin{aligned}
\nabla f(\mathbf{x}) &= \frac{1}{\rho}\left(\nabla(\rho f(\mathbf{x})) - f(\mathbf{x})\nabla\rho\right) && (2.38)\\
\langle\nabla f\rangle_i &= \frac{1}{\rho_i}\left(\sum_{j=1}^{N}\rho_j f_j \frac{m_j}{\rho_j}\nabla_i W_{ij} - f_i\sum_{j=1}^{N}\rho_j\frac{m_j}{\rho_j}\nabla_i W_{ij}\right) && (2.39)\\
\langle\nabla f\rangle_i &= \frac{1}{\rho_i}\sum_{j=1}^{N} m_j\left(f_j - f_i\right)\nabla_i W_{ij} && (2.40)
\end{aligned}$$

Equation 2.40 is another form of computing the gradient (or divergence if f is a vector).

Consider now the identity obtained from the quotient rule of derivatives. Following the same procedure as the previous derivation, one can find another representation of the gradient.

$$\begin{aligned}
\nabla\left(\frac{f(\mathbf{x})}{\rho}\right) &= \frac{\rho\nabla f(\mathbf{x}) - f(\mathbf{x})\nabla\rho}{\rho^2} && (2.41)\\
\nabla f(\mathbf{x}) &= \rho\left(\nabla\left(\frac{f}{\rho}\right) + \frac{f}{\rho^2}\nabla\rho\right) && (2.42)\\
\langle\nabla f\rangle_i &= \rho_i\left(\sum_{j=1}^{N}\frac{f_j}{\rho_j}\frac{m}{\rho_j}\nabla W_{ij} + \frac{f_i}{\rho_i^2}\sum_{j=1}^{N}\rho_j\frac{m}{\rho_j}\nabla W_{ij}\right) && (2.43)\\
\langle\nabla f\rangle_i &= \rho_i\sum_{j=1}^{N} m_j\left(\frac{f_i}{\rho_i^2} + \frac{f_j}{\rho_j^2}\right)\nabla W_{ij} && (2.44)
\end{aligned}$$

Equations 2.40 and 2.44 provide an alternative formulation for the gradient of a function. The alternative forms are more commonly used in physical simulations because the particle function values appear in pairs. Equation 2.44 is anti-symmetric for the interchange of particle indicies, that is $\langle \nabla f \rangle_{ij} = - \langle \nabla f \rangle_{ji}$. If ∇f represents a force in a physical simulation, then Equation 2.44 will obey Newton's Third Law [10].

2.5 Laplacian Approximation

The second derivative or the Laplacian approximation in SPH can be determined by replacing $f(\mathbf{x})$ in Equation 2.17 with $\nabla f(\mathbf{x})$, applying the product rule and invoking the divergence theorem and the compact support property to obtain

$$\left\langle \nabla^2 f \right\rangle = \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} \nabla_i^2 W_{ij} \tag{2.45}$$

The problem with the above expression is that the second derivative of the smoothing function is very sensitive to the particle disorder, that is if the particles are not laid out in a structured way, it can cause unpredictable behaviour due to the fact that $\nabla^2 W$ can have two turning points within its range of compact support. Using this equation is adequate for systems where there is no motion, but for fluid systems where the particles can move, using this equation can cause unrealistic motion.

An alternative to using the second derivative of the kernel is to use Equation 2.30 and replace the function f with a finite difference approximation for the gradient of f.

$$\begin{aligned}
\left\langle \nabla^2 f \right\rangle &= \langle \nabla \cdot \nabla f \rangle = \frac{1}{\rho_i}\sum_{j=1}^{N} m_j \left(\left[\frac{\partial f}{\partial r}\nabla r\right]_j - \left[\frac{\partial f}{\partial r}\nabla r\right]_i \right) \nabla W_{ij} && (2.46)\\
&= \frac{1}{\rho_i}\sum_{j=1}^{N} m_j \left(\frac{f_{ji}}{|\mathbf{r_{ji}}|}\frac{\mathbf{r_{ji}}}{|\mathbf{r_{ji}}|} - \frac{f_{ji}}{|\mathbf{r_{ji}}|}\frac{\mathbf{r_{ij}}}{|\mathbf{r_{ij}}|} \right) \nabla W_{ij} && (2.47)\\
&= \frac{1}{\rho_i}\sum_{j=1}^{N} 2m_j \left(\frac{f_{ij}}{|\mathbf{r_{ij}}|^2} \right) \mathbf{r_{ij}} \nabla W_{ij} && (2.48)\\
&= \frac{1}{\rho_i}\sum_{j=1}^{N} 2m_j \left(\frac{f_{ij}}{|\mathbf{r_{ij}}|^2} \right) \mathbf{r_{ij}} \frac{\partial W_{ij}}{\partial r}\frac{\mathbf{r_{ij}}}{|\mathbf{r_{ij}}|} && (2.49)\\
\left\langle \nabla^2 f \right\rangle &= \frac{1}{\rho_i}\sum_{j=1}^{N} 2m_j \left(f_i - f_j \right) \frac{\partial W_{ij}}{\partial r}\frac{1}{|\mathbf{r_{ij}}|} && (2.50)
\end{aligned}$$

Using the above expression for the Laplacian removes the use of the second derivative of the smoothing function, which allows for better stability [3].

2.6 Corrected Function and Gradient Approximation

Consider the following one-dimensional Taylor series expansion of the function f around the position of particle i, x_i, and evaluated at the position of particle j, x_j.

$$\begin{aligned}
f(x_j) &= f(x_i) + f'(x_i)(x_j - x_i) && (2.51)\\
f(x_j)W_{ij} &= f(x_i)W_{ij} + f'(x_i)(x_j - x_i)W_{ij} && (2.52)\\
\int_\Omega f(x_j)W_{ij}dV_j &= \int_\Omega f(x_i)W_{ij}dV_j + \int_\Omega f'(x_i)(x_j - x_i)W_{ij}dV_j && (2.53)
\end{aligned}$$

Now, ignoring the term with the derivative, $f'(x_i)$ in Equation 2.53 we can solve for the function value at particle i, $f(x_i)$.

$$\begin{aligned}
\int_\Omega f(x_j)W_{ij}dV_j &= \int_\Omega f(x_i)W_{ij}dV_j && (2.54)\\
\frac{\int_\Omega f(x_j)W_{ij}dV_j}{\int_\Omega W_{ij}dV_j} &= f(x_i) && (2.55)
\end{aligned}$$

Replacing the integral with the summation approximation, we are left with a corrected particle approximation for particle *i*.

$$f(x_i) = \frac{\sum_j^N f(x_j) W_{ij} \frac{m_j}{\rho_j}}{\sum_j^N W_{ij} \frac{m_j}{\rho_j}} \tag{2.56}$$

In the SPH notation, the above equation becomes:

$$\langle f \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} f_j W_{ij}}{\sum_j^N \frac{m_j}{\rho_j} W_{ij}} \tag{2.57}$$

Equation 2.57 is the corrected particle function approximation. The benefit of using this function is that the denominator will always normalize the numerator when f_j is equal to 1. Also, since the smoothing function, W, appears in the numerator and the denominator, the smoothing function does not need to be normalized (assuming that the smoothing length, h, is the same for all particles).

Similarly, instead of multiplying the Taylor series expansion by the smoothing function, it was multiplied by the derivative of the smoothing function and ignoring second order derivatives and above, we can obtain a corrected approximation for the gradient of the function.

$$\begin{aligned}
f(x_j) &= f(x_i) + f'(x_i)(x_i - x_j) + f''(x_i)(x_i - x_j)^2 && (2.58)\\
f(x_j)\frac{\partial W}{\partial x} &= f(x_i)\frac{\partial W}{\partial x} + f'(x_i)(x_i - x_j)\frac{\partial W}{\partial x} && (2.59)\\
\int_\Omega f(x_j)\frac{\partial W}{\partial x} &= \int_\Omega f(x_i)\frac{\partial W}{\partial x} + f'(x_i)\int_\Omega (x_j - x_i)\frac{\partial W}{\partial x} && (2.60)\\
f'(x_i) &= \frac{\int_\Omega f(x_j) - f(x_i)\frac{\partial W}{\partial x}}{\int_\Omega (x_j - x_i)\frac{\partial W}{\partial x}} && (2.61)
\end{aligned}$$

Using the standard SPH notation, we are left with a corrected particle approximation for the gradient of a function.

$$\langle f_x \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} \left(f_i - f_j\right) \frac{\partial W_i}{\partial x}}{\sum_j^N \frac{m_j}{\rho_j} \left(x_i - x_j\right) \frac{\partial W_i}{\partial x}} \tag{2.62}$$

The above equation is the corrected particle approximation for the x-derivative of the function and particle i, f_{x_i}. The derivative of the smoothing function is taken with respect to the position of particle i. The y and z derivatives can be obtained in a similar fashion:

$$\langle f_y \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} \left(f_i - f_j\right) \frac{\partial W_i}{\partial y}}{\sum_j^N \frac{m_j}{\rho_j} \left(y_i - y_j\right) \frac{\partial W_i}{\partial y}} \tag{2.63}$$

$$\langle f_z \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} \left(f_i - f_j\right) \frac{\partial W_i}{\partial z}}{\sum_j^N \frac{m_j}{\rho_j} \left(z_i - z_j\right) \frac{\partial W_i}{\partial z}} \tag{2.64}$$

2.7 Summary and List of Notations

The following is list of equations derived in this chapter:

1. Difference notation for a particle property, A. Here A can be any value associated with a specific particle such as position, velocity, density, etc.

$$A_{ij} = A_i - A_j \tag{2.65}$$

2. Density of particle

$$\langle \rho \rangle_i = \sum_{j=1}^N m_j W\left(\mathbf{r}_i - \mathbf{r}_j, h\right) \tag{2.66}$$

3. Particle approximation of a function at a specific particle *i*

$$\langle f \rangle_i = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j W_{ij} \tag{2.67}$$

4. Particle approximation of the gradient of a function at a specific particle i

$$\langle \nabla f \rangle_i = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \nabla_i W_{ij} \tag{2.68a}$$

$$\langle \nabla f \rangle_i = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j \left(f_j - f_i \right) \nabla_i W_{ij} \tag{2.68b}$$

$$\langle \nabla f \rangle_i = \rho_i \sum_{j=1}^{N} m_j \left(\frac{f_i}{\rho_i^2} + \frac{f_j}{\rho_j^2} \right) \nabla_i W_{ij} \tag{2.68c}$$

5. Laplacian of a function

$$\left\langle \nabla^2 f \right\rangle_i = \frac{1}{\rho_i} \sum_{j=1}^{N} 2m_j \left(f_i - f_j \right) \frac{\partial W_{ij}}{\partial r} \frac{1}{|\mathbf{r_{ij}}|} \tag{2.69}$$

6. Gradient of the smoothing function

$$\nabla_i W_{ij} = \left(\frac{\partial W(|\mathbf{r}_{ij}|)}{\partial r} \cdot \frac{1}{|\mathbf{r}_{ij}|} \right) \mathbf{r}_{ij} \tag{2.70}$$

7. Corrected particle function approximation

$$\langle f \rangle_i = \frac{\sum_{j}^{N} \frac{m_j}{\rho_j} f_j W_{ij}}{\sum_{j}^{N} \frac{m_j}{\rho_j} W_{ij}} \tag{2.71}$$

8. Corrected gradient

$$\langle f_x \rangle_i = \frac{\sum_{j}^{N} \frac{m_j}{\rho_j} f_{ij} \frac{\partial W_i}{\partial x}}{\sum_{j}^{N} \frac{m_j}{\rho_j} x_{ij} \frac{\partial W_i}{\partial x}} \tag{2.72a}$$

$$\langle f_y \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} f_{ij} \frac{\partial W_i}{\partial y}}{\sum_j^N \frac{m_j}{\rho_j} y_{ij} \frac{\partial W_i}{\partial y}} \tag{2.72b}$$

$$\langle f_z \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} f_{ij} \frac{\partial W_i}{\partial z}}{\sum_j^N \frac{m_j}{\rho_j} z_{ij} \frac{\partial W_i}{\partial z}} \tag{2.72c}$$

Chapter 3

Discretization of the Navier-Stokes Equations

The Navier-Stokes equations are a set of equations that describe the motion of fluid material. Fluids are substances whose shape deforms under applied shear stress. Solids, on the other hand are capable of resisting shear stress and maintaining its shape and volume. The equations of fluid flow are a continuum formulation of Newton's law of conservation of momentum in which the motion of an infinitesimal volume of mass is directed by the external forces acting on the volume.

In Eulerian form, the *momentum equation* has the form

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla P + \nabla \cdot \boldsymbol{\tau} + \mathbf{f}, \tag{3.1}$$

where $\mathbf{v}$ is the velocity of the fluid, P is the pressure within the fluid, $\boldsymbol{\tau}$ is the deviatoric stress tensor, and $\mathbf{f}$ is the sum of any external forces acting on the fluid. The term $\frac{\partial \mathbf{v}}{\partial t}$ on the left side of the momentum equation represents the acceleration of a fluid volume at a fixed location in space. This form of the momentum equation is usually solved when grid-based methods are used. Instead of using the partial derivatives, the material derivative can be used. The material derivative of a continuous field is

$$\frac{Df}{Dt} = \left(\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f\right). \tag{3.2}$$

The material derivative represents the rate of change of the volume as it moves with the velocity field. Using this in the momentum equation gives us the Lagrangian form for the conservation of momentum of a fluid volume

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla P + \nabla \cdot \boldsymbol{\tau} + \mathbf{f}. \tag{3.3}$$

In SPH, since the fluid is represented by a finite number of particles that move through space, the material derivative is a better choice to use. The momentum equation that we will be investigating is

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla \cdot \boldsymbol{\tau} - \frac{\sigma\kappa}{\rho}\frac{\mathbf{n}}{|\mathbf{n}|}\delta_s + \mathbf{g}, \tag{3.4}$$

where the external force has been split into two terms, the acceleration due to gravity and the surface tension force. Here $\mathbf{g}$ is the gravitational acceleration, σ is the surface tension coefficient, κ is the curvature of the fluid's surface and $\mathbf{n}$ is the outward surface normal of the fluid surface. The first term on the right side of the equation is the acceleration due to the pressure in the fluid; the second term is the acceleration due to viscous forces within the fluid; the third and fourth terms are the external forces of surface tension and gravity. Equation 3.4 is simply a Newton's Second Law equation for the motion of the fluid. The left side of the equation is the acceleration of a fluid mass, and the right side is the sum of all the forces acting on the fluid divided by an inertial property, density.

Aside from the momentum equation, fluids are required to conserve mass during their motion. If we consider a small volume of fluid, as fluid from outside the volume flows into the volume, the density of the fluid volume should increase. Likewise, the density should decrease as fluid flows out of the volume. This behaviour is modeled using the *continuity equation*

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{3.5}$$

or in Lagrangian form:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} \tag{3.6}$$

The momentum equation (Equation 3.1) together with the continuity equation (Equation 3.5) from the generalized Navier-Stokes equation for fluid flow.

For fluids that can be compressed, an equation of state is required to close the system and relate the pressure within the fluid to other properties of the system. An example of an equation of state is the ideal gas equation. For most liquids, the density changes in the fluid flow are minuscule. Such fluids are known as incompressible fluids. Incompressible fluids are modeled so that the material derivative of its density is zero. This enforces a divergence free condition for the velocity field of the fluid.

$$\nabla \cdot \mathbf{v} = 0 \tag{3.7}$$

We discretize the Navier-Stokes equations into a set of ordinary differential equations by discretizing each of the derivatives and gradients that appear in the equations according to the SPH formulations. The pressure and the viscous forces are discretized using Equation 2.44 and the continuity equation is discretized using Equation 2.40, yielding:

$$\left\langle \frac{\nabla P}{\rho} \right\rangle_i = \sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W_{ij}, \tag{3.8}$$

$$\left\langle \frac{\nabla \cdot \boldsymbol{\tau}}{\rho} \right\rangle_i = \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\tau}_i}{\rho_i^2} + \frac{\boldsymbol{\tau}_j}{\rho_j^2} \right) \nabla W_{ij}, \text{ and} \tag{3.9}$$

$$\langle \rho \nabla \cdot \mathbf{v} \rangle_i = -\sum_{j=1}^{N} m_j \left(\mathbf{v_j} - \mathbf{v_i} \right) \cdot \nabla W_{ij}. \tag{3.10}$$

Combining these three equations with the Navier-Stokes equations results in a set of ordinary differential equations to solve for each particle.

$$\begin{aligned}\left\langle \frac{D\mathbf{v}}{Dt} \right\rangle_i &= -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W_{ij} \\ &\quad + \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\tau}_i}{\rho_i^2} + \frac{\boldsymbol{\tau}_j}{\rho_j^2} \right) \nabla W_{ij} + \mathbf{f}_{ext} \qquad (3.11)\\ \left\langle \frac{D\mathbf{r}}{Dt} \right\rangle_i &= \mathbf{v_i} \qquad (3.12)\end{aligned}$$

$$\left\langle \frac{D\rho}{Dt} \right\rangle_i = \sum_{j=1}^{N} m_j \left(\mathbf{v_i} - \mathbf{v_j} \right) \cdot \nabla W_{ij} \qquad (3.13)$$

The density evolution describe by the continuity equation may be omitted if the fluid flow is to be incompressible. See Section 3.3 for incompressible fluid flow.

3.1 Modelling Stress

Viscosity is the internal friction of the fluid and is responsible for dissipating the energy within the fluid. The viscosity is represented as the divergence of the deviatoric stress tensor,$\boldsymbol{\tau}$, in Equation 3.4.

Newtonian fluids are fluid in which the stress tensor is linearly proportional to the rate of strain tensor, **D**. That is

$$\boldsymbol{\tau} = \mu \mathbf{D}, \qquad (3.14)$$

where μ is the dynamic viscosity coefficient and the rate of strain tensor for incompressible fluids is given by

$$\mathbf{D} = \frac{1}{2}\nabla \mathbf{v} + \frac{1}{2}\left(\nabla \mathbf{v}\right)^T = \begin{bmatrix} \partial_x u & \frac{1}{2}\left(\partial_y u + \partial_x v\right) & \frac{1}{2}\left(\partial_z u + \partial_x w\right) \\ \frac{1}{2}\left(\partial_y u + \partial_x v\right) & \partial_y v & \frac{1}{2}\left(\partial_z v + \partial_y w\right) \\ \frac{1}{2}\left(\partial_z u + \partial_x w\right) & \frac{1}{2}\left(\partial_z v + \partial_y w\right) & \partial_z w \end{bmatrix}. \quad (3.15)$$

The variables u, v and w are the Cartesian components of the velocity vector.

For incompressible Newtonian fluids, the divergence of the stress tensor simplifies to the Laplacian of the velocity field.

$$\nabla \cdot \boldsymbol{\tau} = \mu \nabla^2 \mathbf{v} \quad (3.16)$$

For non-Newtonian fluids the stress tensor is not linearly proportional to the rate of strain tensor. The viscosity coefficient, μ can be a function of the local properties of the fluid such as temperature, or more commonly the rate of strain. Many models exist to model the viscosity for non-Newtonian fluids. Barnes et al. showed that the viscosity of non-Newtonian fluids can be effectively modeled by the Cross Model [17] [1]:

$$\mu(\gamma) = \mu_\infty + \frac{\mu_0 - \mu_\infty}{1 + C\gamma^a}, \quad (3.17)$$

where μ_0 and μ_∞ are the dynamic viscosities at low and high shear rates and γ is the Frobenius norm of $\mathbf{D}$ defined by:

$$\gamma = |\mathbf{D}| = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3} |D_{ij}|^2} \quad (3.18)$$

The parameters C and a that appear in Equation 3.17 describe the qualitative rate of change of viscosity with respect to changes in the magnitude of the internal stresses in the fluid (see Figure 3.1). The parameters essentially define the location and rate of change of the transition between the two bounds on viscosity.

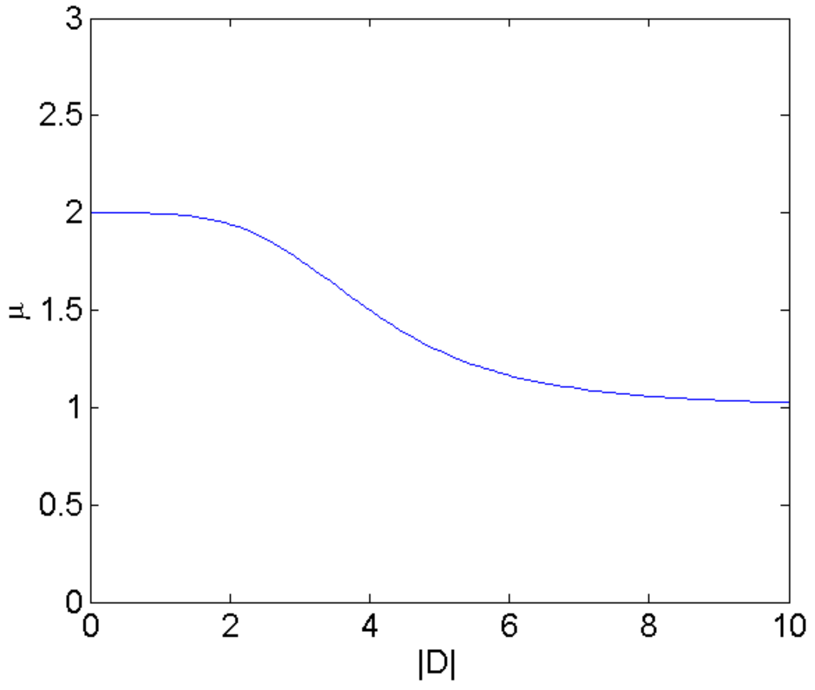


Figure 3.1: The Cross model for non-Newtonian viscosity.

The benefit of the Cross model is that the viscosity is defined for all values of γ and is bounded by the two extremes, μ_∞ and μ_0. The drawback is that there are four parameters, μ_∞, μ_0, C and a, in the model that need to be determined to accurately model the non-Newtonian fluid. Other viscosity models exist, but some models lead to unphysical predictions when the computed viscosity vanishes or blows up at finite stresses (e.g., see [8]).

To compute the viscous forces acting on a particle, the rate of strain tensor must be constructed for the particle. The partial derivatives in the rate of strain tensor are computed using the corrected gradient formula, Equations 2.72a - 2.72c. For example, we compute $\frac{\partial u}{\partial x}$ by the following equation, the rest of the partial derivatives can be computed similarly.

$$\langle u_x \rangle_i = \frac{\sum_j^N \frac{m_j}{\rho_j} u_{ij} \frac{\partial W_{ij}}{\partial x}}{\sum_j^N \frac{m_j}{\rho_j} x_{ij} \frac{\partial W_{ij}}{\partial x}} \tag{3.19}$$

$$= \frac{\sum_j^N \frac{m_j}{\rho_j} (u_i - u_j) \left(\frac{\partial W_{ij}}{\partial r} \frac{1}{|r_{ij}|} \right) (x_i - x_j)}{\sum_j^N \frac{m_j}{\rho_j} (x_i - x_j) \left(\frac{\partial W_{ij}}{\partial r} \frac{1}{|r_{ij}|} \right) (x_i - x_j)} \tag{3.20}$$

Once the rate of strain tensor is constructed, the viscosity coefficient for each particle is calculated using Equation 3.17 and the stress tensor is constructed by multiplying the viscosity coefficient with the rate of strain tensor. The viscous accelerations can then be computed using Equation 3.9.

3.2 Modelling Surface Tension

Surface tension plays an important role in droplet formation. Surface tension at the microscopic level is created by the cohesion of the molecules within the fluid. In the interior region of a volume of fluid, individual molecules experience cohesive forces from interactions with neighbouring molecules in all directions. The total of all the cohesive forces acting on any given molecule in the interior of the fluid is zero on average. At the fluid's external boundary, however, there are no neighbouring fluid molecule interactions so the cohesion of molecules results in a net force acting on the fluid surface. If the fluid interface is curved towards the fluid, the net force due to the cohesive forces produces a net force that points towards the fluids and is proportional to the curvature of the fluid interface. If the interface is curved away from the fluid, the net force is in the opposite direction. Mathematically the surface tension force is represented as:

$$\mathbf{F_s}(\mathbf{r}) = -\sigma\kappa\frac{\mathbf{n}}{|\mathbf{n}|}\delta_s\ , \tag{3.21}$$

where κ is the curvature of the fluid interface, σ is the surface tension coefficient, $\mathbf{n}$ is the outward surface normal of the fluid interface, and δ_s is the Dirac Delta function whose argument is zero on the fluid surface. The surface tension in our experiments follows the work done by Zhang[19].

3.2.1 Identifying Surface Particles in Two-Dimensions

In actual fluid simulations, to compute the surface tension at the fluid interface,the specific location of the fluid interface must be known. That is, it is necessary to identify the particles of fluid that lie in the interior of the fluid volume and the particles that lie at the interface of the fluid and the outside region. We identify surface particles using a method developed by Dilts[4].

We illustrate the computational method of Dilts by first considering a fluid in two dimensions. Consider a particle, i, surrounded by a number of other particles, j, in close proximity. An imaginary circle with a radius equal to the initial particle spacing is created at the location of each particle. If the circle of particle i is fully covered by the circles of its neighbour particles then particle i is tagged as an interior fluid particle. If particle i's circle is not fully covered, then particle i is tagged as a surface particle.

Let $\mathbf{r_i}$ and $\mathbf{r_j}$ be the centers of circles i and j, and R_i and R_j be the radii of each circle.

- If d = $\left|\mathbf{r}_j - \mathbf{r}_i\right| \geq R_i + R_j$, then the two circles intersect at one point or do not intersect at all.
- If d = $\left|\mathbf{r}_j - \mathbf{r}_i\right| < R_i + R_j$, then the two circles intersect at two points.

If the two circles intersect at two points, then the location of the two points of intersection is given by the following steps (see Figure 3.2 for variable references):

1.
$$a = \frac{R_i^2 - R_j^2 + d^2}{2d} \tag{3.22}$$

2.
$$h = \sqrt{R_i^2 - a^2} \tag{3.23}$$

3.
$$\mathbf{r_c} = \mathbf{r_i} + \frac{a}{d}\left(\mathbf{r_j} - \mathbf{r_i}\right) \tag{3.24}$$

4.
$$x_1 = x_c + \frac{h}{d}\left(y_j - y_i\right) \tag{3.25}$$
$$y_1 = y_c - \frac{h}{d}\left(x_j - x_i\right) \tag{3.26}$$
$$x_2 = x_c - \frac{h}{d}\left(y_j - y_i\right) \tag{3.27}$$
$$y_2 = y_c + \frac{h}{d}\left(x_j - x_i\right) \tag{3.28}$$

Given two points $\mathbf{r_1} = (x_1, y_1)$ and $\mathbf{r_2} = (x_2, y_2)$ on circle i, the angle that each point subtends from the positive x-axis is:

$$\alpha_1 = \text{atan2}(y_1 - y_i, x_1 - x_i) \tag{3.29}$$
$$\alpha_2 = \text{atan2}(y_2 - y_i, x_2 - x_i) \tag{3.30}$$

We must make sure that each of these angles are in the range $[0, 2\pi]$, we do this by the following formula:

$$\alpha = \begin{cases} \alpha, & \alpha > 0 \\ \alpha + 2\pi, & \alpha < 0 \end{cases} \tag{3.31}$$

After both angles are readjusted to lie between $[0, 2\pi]$, we must determine whether circle j covers the arc $[\alpha_1, \alpha_2]$, or $[\alpha_2, 2\pi] \bigcup [0, \alpha_1]$ (assuming that $\alpha_1 < \alpha_2$). To determine

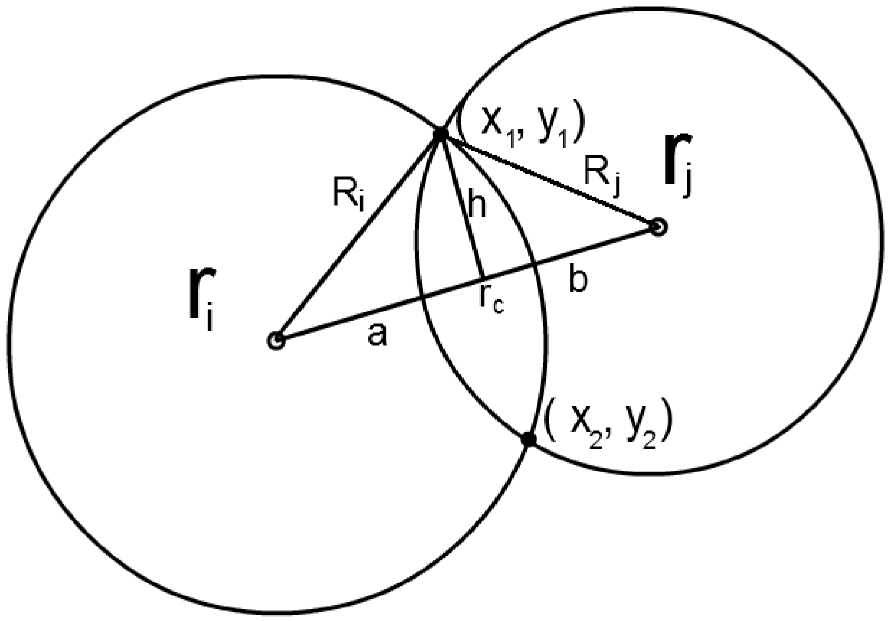


Figure 3.2: The intersection of two circles.

this, the angle the distance vector $\mathbf{d} = \mathbf{r_j} - \mathbf{r_i}$ makes with the positive x-axis is checked to see in which range the angle falls in. The angle the distance vector makes with the postive x-axis is

$$\mathbf{d} = \mathbf{r_j} - \mathbf{r_i} \tag{3.32}$$

$$\alpha_c = \begin{cases} \text{atan2}(d_y, d_x), & \text{atan2}(d_y, d_x) > 0 \\ \text{atan2}(d_y, d_x) + 2\pi, & \text{atan2}(d_y, d_x) < 0 \end{cases} \tag{3.33}$$

The interval, I_j, that circle j covers is then:

$$I_j = \begin{cases} [\alpha_1, \alpha_2], & \alpha_c \in [\alpha_1, \alpha_2] \\ [\alpha_2, 2\pi] \bigcup [0, \alpha_1], & \text{Otherwise} \end{cases} \tag{3.34}$$

The union of all such intervals of all neighbouring circles of particle i determines whether particle i is a surface particle or an interior fluid particle. If equation 3.35 is satisfied, then particle i is an interior fluid particle. If it is not satisfied, then particle i is a surface particle.

$$\bigcup_{j=1}^{N} I_j = [0, 2\pi] \tag{3.35}$$

3.2.2 Calculating Surface Tension in Two Dimensions

Once all the surface particles have been found, a local reconstruction of the fluid surface can be performed to calculate the curvature and surface normal of the fluid interface. Consider for the moment a surface particle, i, surrounded by its neighbouring particles (interior and surface particles), j. A rotated coordinate system is constructed whose y-axis passes through the center of mass of particle i's support domain. We find the direction of the coordinate system by

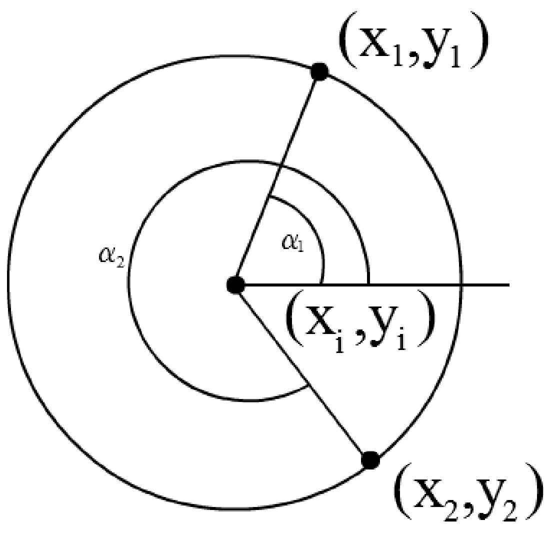


Figure 3.3: Arcs that each intersection point makes with the positive x-axis.

$$\widehat{\mathbf{y}} = \mathbf{r_i} - \frac{\sum_j^n r_j}{n} \tag{3.36}$$

$$\widehat{\mathbf{x}} = \left[-\widehat{y}_y, \widehat{y}_x\right] \tag{3.37}$$

where n is the number of particles within particle i's support domain. The rotated coordinate axes should be normalized:

$$\widehat{\mathbf{x}} \leftarrow \frac{\widehat{\mathbf{x}}}{|\widehat{\mathbf{x}}|} \tag{3.38}$$

$$\widehat{\mathbf{y}} \leftarrow \frac{\widehat{\mathbf{y}}}{|\widehat{\mathbf{y}}|} \tag{3.39}$$

After the rotated coordinate system is constructed, we project the position vectors of neighbouring *surface particles*, k, of particle i onto the rotated coordinate system:

$$x_j = (\mathbf{r}_k - \mathbf{r_i}) \cdot \widehat{\mathbf{x}} \tag{3.40}$$

$$y_j = (\mathbf{r}_k - \mathbf{r_i}) \cdot \widehat{\mathbf{y}} \tag{3.41}$$

The set of new coordinates of all surface particles near particle i, including particle i itself whose coordinates in the rotated system will be $(0, 0)$, are used to construct a Vandermonde system for a quadratic curve. For n surface particles the Vandermonde system looks like

$$\begin{bmatrix} 0 & 0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.42}$$

This system is an overdetermined system and is in the form

$$\mathbf{Ac} = \mathbf{b}, \tag{3.43}$$

The coefficient vector $\mathbf{c}^{\mathbf{T}} = [a, b, c]$ then produces the local parabola at particle i in the rotated coordinate system and is of the form

$$y = ax^2 + bx + c \tag{3.44}$$

The overdetermined system will unlikely have any solutions, so the best approximate solution must be solved using the normal equations, that is we solve the following equation instead:

$$\left(\mathbf{A}^{\mathbf{T}}\mathbf{A}\right)\mathbf{c} = \mathbf{A}^{\mathbf{T}}\mathbf{b} \tag{3.45}$$

The curvature of the fluid interface is found by calculating the curvature of the above parabola at $x = 0$. The general equation for the curvature of a function is

$$\kappa = \frac{y''}{(1 + y')^{3/2}} \tag{3.46}$$

which, for $x = 0$ simplifies to

$$\kappa = \frac{2a}{(1 + b)^{3/2}}. \tag{3.47}$$

The direction of the surface normal is then given by

$$\begin{aligned} \widehat{\mathbf{n}} &= \left[y'(0), -1\right] = [b, -1] && (3.48) \\ \widehat{\mathbf{n}} &\leftarrow \frac{\widehat{\mathbf{n}}}{|\widehat{\mathbf{n}}|} && (3.49) \end{aligned}$$

The normal vector must then be normalized as given in equation 3.49. This surface normal vector is in the rotated system and must be rotated back into the original coordinate system by

$$\mathbf{n} = \widehat{n}_x\widehat{\mathbf{x}} + \widehat{n}_y\widehat{\mathbf{y}} \tag{3.50}$$

Equations 3.50 and 3.47 give the surface normal and curvature of the fluid interface for a single surface particle. Once the surface normals and curvature for all surface particles have been found, the surface tension force at each particle can then be calculated by

$$\mathbf{f_s} = -\sigma \widehat{\mathbf{n}}_i \kappa_i \tag{3.51}$$

3.3 Modelling Pressure

The pressure of a gaseous fluid is handled differently than a liquid fluid. Gases are a compressible fluid which means its density can change depending on certain conditions. For gases, the Ideal Gas Law governs the pressure inside a gas volume. The Ideal Gas Law is usually written as:

$$PV = nk_bT, \tag{3.52}$$

where V is the volume of the gas, n is the number of particles in the gas, k_b is Boltzmann's Constant and T is the temperature of the gas. The Ideal Gas Law states that the pressure of the fluid is proportional to the number density of the gas, $\frac{n}{V}$. If a gas is to be modeled with SPH, the pressure of the fluid is determined using the following equation of state that is linearly proportional to the density:

$$P_i = C\rho_i, \tag{3.53}$$

where C is a constant that is usually proportional to the square of the speed of sound in the gas. The density is found using the summation density

$$\langle \rho \rangle_i = \sum_{j=1}^{N} m_j W_{ij} \tag{3.54}$$

From the density, the pressure is calculated using Equation 3.53 and the pressure gradient in the Navier-Stokes equation is computed using Equation 3.8.

Liquids, on the other hand, do not compress easily and are considered incompressible for practical purposes. There are two ways to handle the incompressibility of fluids in SPH. One method is to use a stiff equation of state that relates the pressure of the fluid to the density. The equation of state that is usually used is Tate's equation and has the form

$$P(\rho) = c_s^2 \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right). \tag{3.55}$$

Here ρ_0 is the rest density of the fluid, c_s is the speed of sound in the fluid and γ is a constant usually taken to be 7 [13] [2]. If Tate's equation is used as the equation of state, then the density of the fluid particles must be time-stepped along with their position and velocity using Equations 3.11-3.13. The density is then used to calculate the pressure of each particle using the equation of state, Equation 3.55. The problem with this method is that to simulate incompressibility, the speed of sound must be fairly large. The speed of sound must be at least 10 times larger than the maximum velocity that is expected in the simulation. This ensures that the deviation in the density of the fluid is only at 1% of the rest density[13]. Using this stiff equation of state requires that the step-size be very small.

The other option is to enforce strict incompressibility by constraining the velocity field to be divergence free,

$$\nabla \cdot \mathbf{v} = 0. \tag{3.56}$$

In fixed grid methods, enforcing the divergence free condition for the velocity field is performed by using the pressure-correction method. The SPH equivalent of the pressure correction method is outlined as follows. Starting from some initial time-step, t; the velocity, $\mathbf{v_i}$ and positions, $\mathbf{r_i}$, of the particles are evolved to an intermediate state using only the viscous and external forces. The pressure force is neglected in this step:

$$\mathbf{v_i}^* = \mathbf{v_i}(t) + \Delta t \left(\left\langle \frac{\nabla \cdot \boldsymbol{\tau}}{\rho} \right\rangle_i + \mathbf{f_s}_i + \mathbf{g} \right) \tag{3.57}$$

$$\mathbf{r_i}^* = \mathbf{r_i}(t) + \Delta t \mathbf{v_i}^* \tag{3.58}$$

At this intermediate time-step, the divergence free condition, equation 3.56, is not satisfied. The pressure at this intermediate time-step is what forces the next time step to be divergence free.

$$\mathbf{v_i}\,(t + \Delta t) = \mathbf{v_i}^* - \Delta t \left(\frac{\nabla P}{\rho} \right) \tag{3.59}$$

By taking the divergence of both sides of the above equation and forcing the divergence of the next time step to be zero, we get the following Pressure-Poisson Equation that must be solved:

$$\nabla \cdot \left(\frac{\nabla P}{\rho} \right) = \frac{\nabla \cdot \mathbf{v}^*}{\Delta t} \tag{3.60}$$

The boundary conditions for the pressure are usually homogeneous Dirichlet boundary conditions on fluid-air interfaces and homogeneous Neumann boundary conditions on fluid-solid surfaces.

In the SPH formulation, Equation 3.60 is discretized according to the SPH Laplacian, Equation 2.69. A slight modification is done to the equation to keep resulting system of equations symmetric. The modification is to replace any occurrence of particle density with the average density between particle i and j [17].

$$\nabla \cdot \left(\frac{\nabla P}{\rho} \right)_i = \sum_{j=1}^{N} \frac{8 m_j}{(\rho_i + \rho_j)^2} \left(P_i - P_j \right) \frac{\partial W_{ij}}{\partial r} \frac{1}{|r_{ij}|} \tag{3.61}$$

The divergence of the intermediate velocity field is obtained using equation 2.68c.

$$\nabla \cdot \mathbf{v}^* = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j \left(\mathbf{v}_i^* - \mathbf{v}_j^* \right) \cdot \nabla W_{ij} \tag{3.62}$$

Once pressure is solved, the position and the velocities of the particles are time-stepped into their new position using the pressure gradient.

$$\begin{aligned} \mathbf{v_i}(t+\Delta t) &= \mathbf{v_i}^* - \Delta t \sum_{j}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W_{ij} && (3.63) \\ \mathbf{r_i}(t+\Delta t) &= \mathbf{r_i}^* + \Delta t \mathbf{v_i}(t+\Delta t) && (3.64) \end{aligned}$$

3.3.1 Solving the Pressure-Poisson Equation

The SPH Pressure-Poisson Equation, equation 3.61, once discretized for each particle has the form:

$$\begin{aligned} P_i \sum_{j=1}^{N} \left(\frac{8m_j}{(\rho_i+\rho_j)^2} \frac{\partial W_{ij}}{\partial r} \frac{1}{r_{ij}} \right) - \sum_{j=1}^{N} \left(P_j \frac{8m_j}{(\rho_i+\rho_j)^2} \frac{\partial W_{ij}}{\partial r} \frac{1}{r_{ij}} \right) && (3.65) \\ = \frac{1}{\Delta t \rho_i} \sum_{j=1}^{N} m_j \left(\mathbf{v}_{ij}^* \right) \cdot \nabla W_{ij} && (3.66) \\ && (3.67) \end{aligned}$$

The resulting system of equations looks as follows:

$$\begin{bmatrix} \sum_{j=1}^{N} A_{1j} & -A_{12} & -A_{13} & \cdots & -A_{1N} \\ -A_{21} & \sum_{j=1}^{N} A_{2j} & -A_{23} & \cdots & -A_{2N} \\ -A_{31} & -A_{32} & \sum_{j=1}^{N} A_{3j} & \cdots & -A_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -A_{N1} & -A_{N2} & -A_{N3} & \cdots & \sum_{j=1}^{N} A_{NN} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix}, \tag{3.68}$$

where

$$A_{ij} = \frac{8m_j}{(\rho_i + \rho_j)^2} \frac{\partial W_{ij}}{\partial r} \frac{1}{r_{ij}} \tag{3.69}$$

$$b_i = \frac{1}{\Delta t \rho_i} \sum_{j=1}^{N} m_j \left(\mathbf{v}_i^* - \mathbf{v}_j^* \right) \cdot \nabla W_{ij} \tag{3.70}$$

The diagonal element in each row of the coefficient matrix is simply the negative of the sum of all the other elements in that row. Since most of the particles are farther than the support radius, h, the coefficient matrix ends up being sparse. The coefficient matrix is positive-definite as well as symmetric, that is $A_{ij} = A_{ji}$ which can be solved using the Conjugate Gradient method [16].

Chapter 4

Numerical Results

4.1 Implementation

4.1.1 Nearest Neighbours Search

The summations in all the SPH equations are summations over all particles in the system. The naive approach to computing the summations is to perform the summation over all indices. Since the kernel functions have compact support, any particles that lie farther than a distance of h will have a value of zero. It is unnecessary to compute the majority of the terms in the summation since most of the terms will equal zero. There are two techniques that are commonly used to reduce the terms in the summation. Both methods involve searching for nearest neighbour pairs that are within a certain Euclidean distance. The two techniques are spatial hashing and binary tree searches.

Spatial hashing involves subdividing the spatial domain into square or cubic cells with side length equal to $2h$. Each cell contains an array of all the particles whose positions lie within that cell. When the SPH summations are performed for a particular particle, i, we determine what spatial cell that particle is in, and extract all the particles that are in that cell, plus the 8 neighbouring cells (26 cells for 3 dimensions). The best implementations of storing and retrieving nearest neighbours can run in $O(kN)$

time where k is usually on the order of the number of particles per cell, and N is the number of particles[10].

The binary tree search involves using a k-dimensional tree structure. The k-d tree is a binary tree in which every node is a k-dimensional point. In our case, it is a 2-dimensional point. Each node represents a hyperplane that splits the domain into two parts. When new points are added to the tree, these new points are added to one of the two new subdivisions and the subdivision that the particle was added to is subdivided once more. The insertion and finding of nearest neighbours usually runs in $O(N \log(N))$ time [7].

In our implementation we chose to implement the k-d tree to increase the speed of the computations. There are an abundant amount of k-d tree codes freely available on the Internet which made finding and implementing one of these library easy to accomplish. Although spatial hashing can be faster than tree searches, we chose to use the k-d tree structure because of its ease of use and implementation. At the beginning of each iteration, the k-d tree is destroyed and the particles reinserted back into the tree. After the particles are inserted into the tree, the nearest neighbours of all particles are found.

The speed of the summations was further increased by multi-threading the computations. Each thread was assigned a number of particles that it was responsible for. When the summations were required, each thread computed the summations for its assigned particles. This allowed the speed of the computation to scale linearly with the number of processing cores available on the machine performing the computation.

Solving of the Pressure Poisson Equation was performed using the Conjugate Gradient Method. We used a freely available linear algebra library called *Eigen* to solve the system of equations. To increase the speed further, the matrix-vector product algorithm was also multi-threaded. Parallelization of the matrix-vector product algorithm was achieved by dividing the matrix $N \times N$ matrix into p smaller matrices, each of which is of size $\frac{N}{p} \times N$, where p is the number of processors available on a particular machine. For example, a machine with 3 processors would perform the following computation

$$\begin{aligned}
\mathbf{Ax} &= \left[\begin{array}{c} \mathbf{B_1} \\ \hline \mathbf{B_2} \\ \hline \mathbf{B_3} \end{array}\right] \mathbf{x} \qquad (4.1) \\
&= \left[\begin{array}{c} \mathbf{B_1 x} \\ \hline \mathbf{B_2 x} \\ \hline \mathbf{B_3 x} \end{array}\right] \qquad (4.2) \\
&= \mathbf{y} \qquad (4.3)
\end{aligned}$$

where $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ are blocks that make up matrix $\mathbf{A}$. Each of the matrix-vector products in Equation 4.2 are performed on separate threads and the resulting vectors are combined to produce vector $\mathbf{y}$. Eigen was also used to solve the normal equations required for finding the surface tension.

4.1.2 Coverage of a Circle

In Section 3.2 we discussed the method to find surface particles from a point cloud. A circle is covered if the union of all the intervals of coverage is equal to the interval $[0, 2\pi]$. That is:

$$\bigcup_{j=1}^{N} I_j = [0, 2\pi] \qquad (4.4)$$

Haque and Dilts provided a method to determine the coverage of a circle using dynamic linked lists [6]. The implemented data structure contains a linked-list of two dimensional points each of which represent a particular closed interval. Rather than determining the union of all the intervals, we start with the interval of $[0, 2\pi]$ and subtract each of the individual intervals of coverage, I_j until the size of the original interval is zero. Each time an interval is subtracted from the list, the interval nodes in the list are split into two nodes if it fully intersects the interval to be subtracted; removed if the node is contained within the subtracted interval; or resized if it partially intersects the subtracted interval. For example, consider the following set operation: $S = [0, 10] - [4, 6] - [2, 4] - [7, 8]$. The evolution of the linked list structure is shown in Table 4.1.

Set Operation	Resulting Linked List
$S := [0, 10]$	$[0, 10] \rightarrow$
$S := S - [5, 6]$	$[0, 4] \rightarrow [6, 10] \rightarrow$
$S := S - [2, 4]$	$[0, 2] \rightarrow [6, 10] \rightarrow$
$S := S - [7, 8]$	$[0, 2] \rightarrow [6, 7] \rightarrow [9, 10] \rightarrow$

Table 4.1: Sixth-order Guassian-like kernel.

We implemented Haque's and Dilts' method in C++ by creating a class that inherits from the linked list data structure provided by the Standard Template Library.

4.2 Convergence Analysis

In section 2.1.1 it was shown that the interpolation of a function using SPH yields $O(h^2)$ accuracy. This section tests this claim. In finite difference schemes, the accuracy of the approximation is usually quantified or measured in relation to Δx, the distance between adjacent grid points. As the distance Δx between adjacent grid nodes decreases, the error of the associated solution computed is asymptotically proportional to some (positive) power of Δx. With SPH, separation distance is not the only factor that affects the accuracy of the approximation. The smoothing length, h, plays an important role in both the accuracy and the stability of the system that it is trying to approximate.

Consider the function

$$f(x) = \sin(\exp(3x)), x \in [0, 1] \tag{4.5}$$

This function, and its derivative are oscillatory functions that have low frequency near the left endpoint of the interval and a higher frequency and amplitude (for the derivative) near the right end of the interval (see the plots of f and f' as shown in Figure 4.1). This function was chosen because it contains both low and high frequency oscillations so that we could investigate the accuracy of the interpolation scheme in both regimes.

To approximate this function, N particles were uniformly distributed in the interval $[0, 1]$. The function and its gradient are then approximated using the corrected particle function approximation, Equation 2.71, and the corrected gradient approximation, Equation 2.72a. These computations are repeated while varying N (the number of particles) and h (the smoothing length) to examine the accuracy of the corresponding approximations. We compare the error in the approximation to the average particle spacing $\Delta x = \frac{N}{L}$, where is the length of the interval, in our case $L = 1$, and the number of nearest neighbours, k, within the smoothing length. That is, the size of the kernel encompasses, k, particles before its value reaches zero.

We use the following formulas to determine the error in the approximations

$$\text{Function Error} = \left[\sum_{i=1}^{N} \left|f(x_i) - \langle f \rangle_i\right|^2 \Delta x\right]^{1/2} \tag{4.6}$$

$$\text{Gradient Error} = \left[\sum_{i=1}^{N} \left|f'(x_i) - \langle f_x \rangle_i\right|^2 \Delta x\right]^{1/2}, \tag{4.7}$$

where $f(x_i)$ and $f'(x_i)$ are the actual function and it's derivative evaluated at the lo-

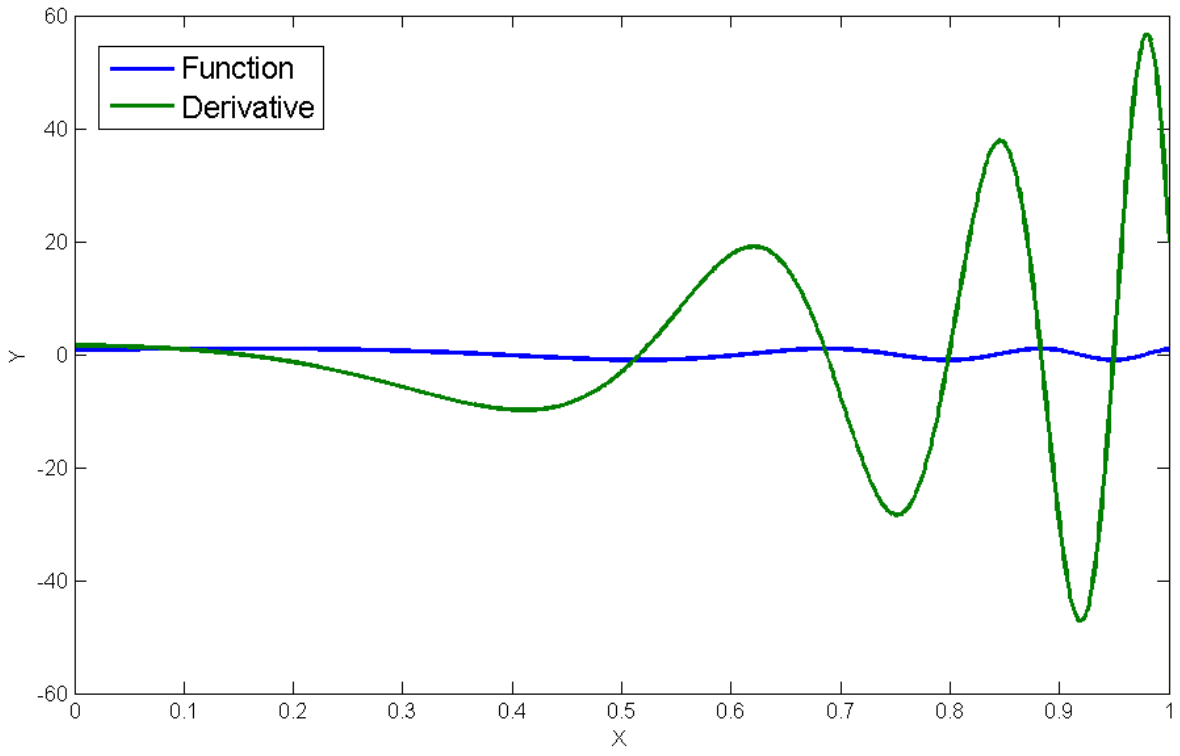


Figure 4.1: Function, $f(x) = \sin(\exp(3x))$, and derivative, $f'(x) = 3\cos(\exp(3x))\exp(3x)$, to be approximated.

cation of particle i, and $\langle f \rangle_i$ and $\langle f_x \rangle_i$ is the function and its gradient calculated using Equation 2.71 and 2.72a and using the sixth order polynomial kernel, Equation 2.18.

In the two error plots, Figure 4.2 and 4.3, the total error decreases as the number of particles increase in the domain. This is equivalent to the decreasing the smoothing length, h, for constant number of nearest neighbours. The slope of each line in figure 4.2 is ≈ -2 which shows the second order accuracy derived in Equation 2.11. The gradient convergence, figure 4.3, on the other hand shows an order of accuracy close to $O(h)$. The low order of accuracy can lead to large errors in numerical simulations.

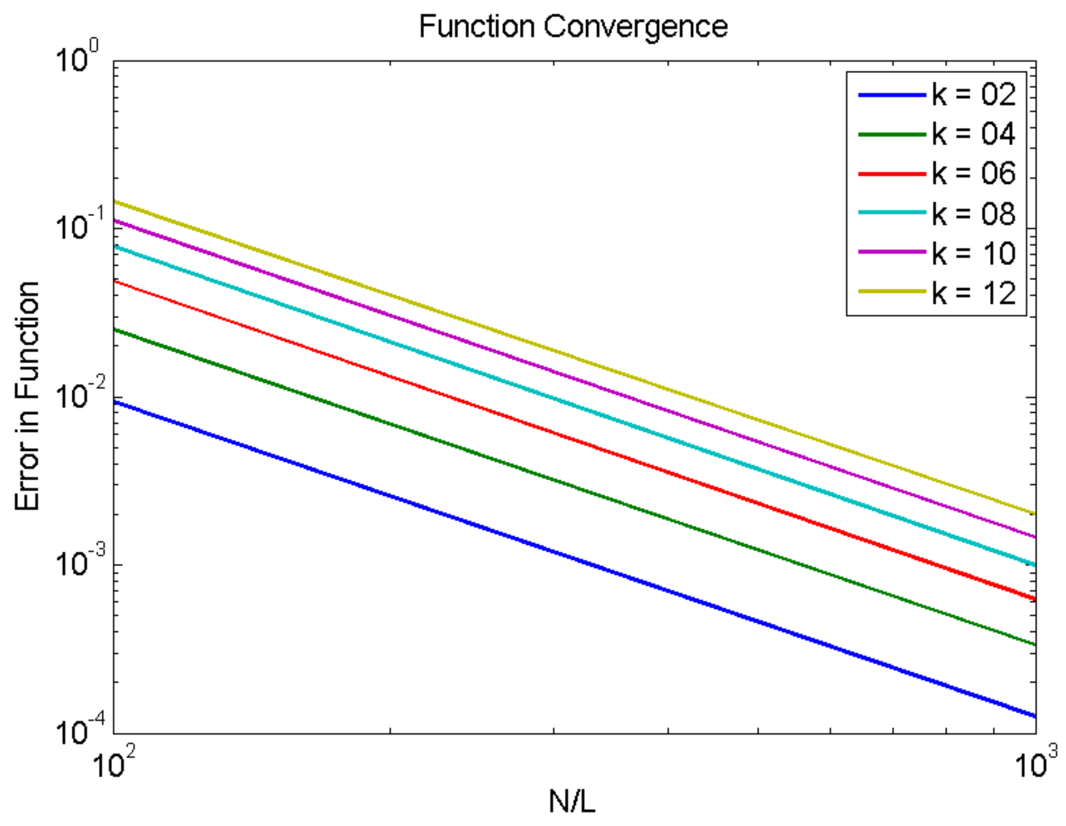


Figure 4.2: Error plot for approximating the function $f(x)$ using the corrected particle function approximation for different number of particles as a function of smoothing length (represented as the number of particles within the smoothing length).

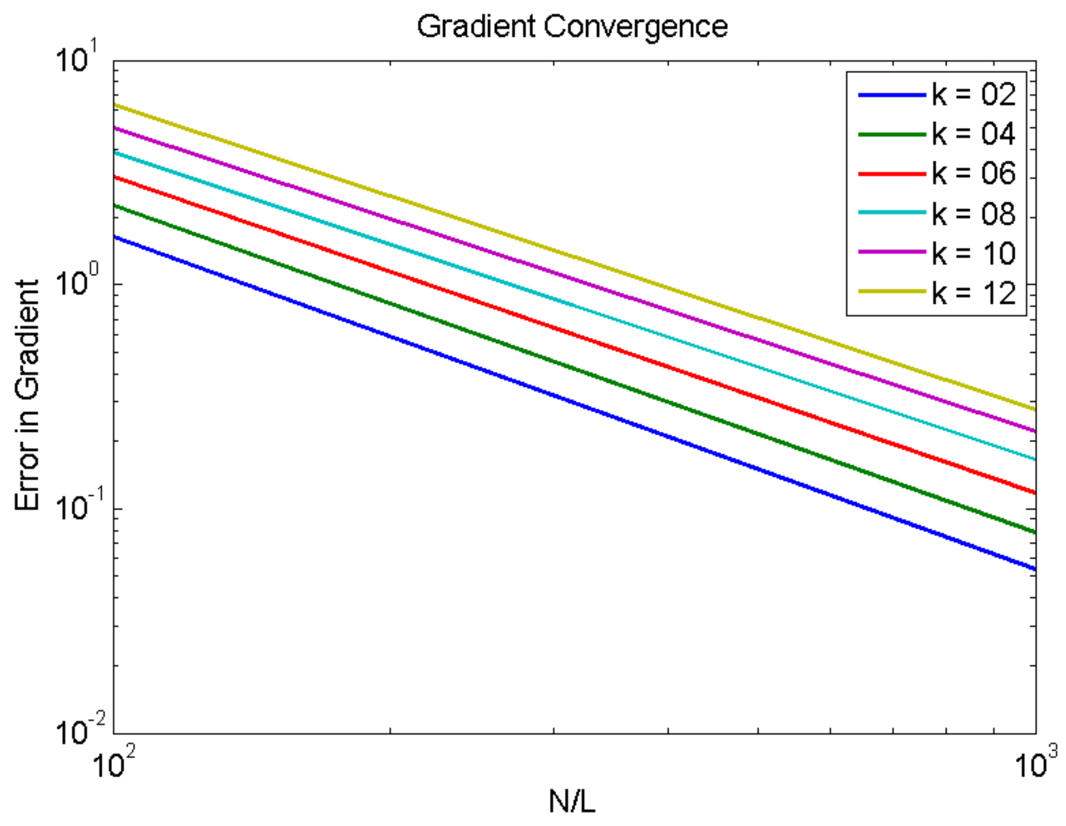


Figure 4.3: Error plot for approximating the derivative of the function $f(x)$ using the corrected particle gradient approximation with various number of particles and smoothing length.

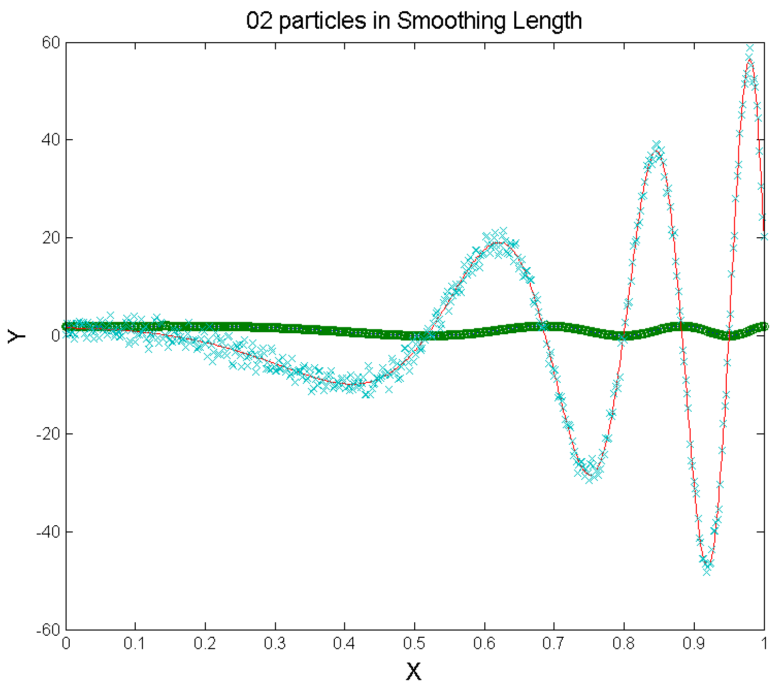
4.2.1 Smoothing Length

In Figures 4.2 and 4.3, the total error is smallest when there are only two particles in the smoothing length. A question arises as to why ever have more than two particles within the smoothing length if two particles yields the smallest error? In the previous case, each particle was given the function value $f_i = f(x_i)$, but consider for the moment, that each particle is then given a small random pertubation of 1% of the function value.

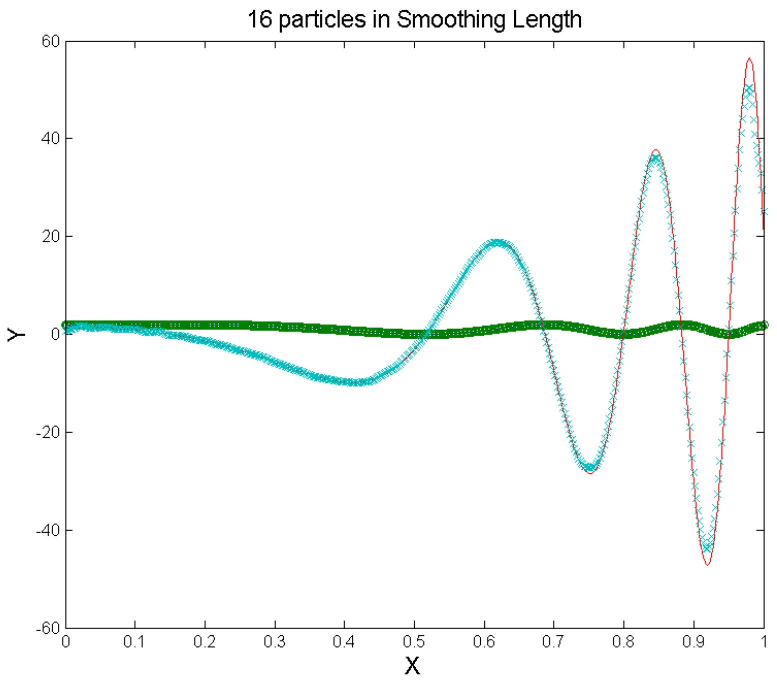
Calculating the function value and the gradient using only two particles in the smoothing length, as was done previously shows that even though the error in the function value is very small, the error in the gradient is very large, especially in areas where the gradient does not change very much. This is not surprising considering numerical differentiation is always sensitive to noise.

Figures 4.4a and 4.4b show the particle function and gradient value for a smoothing scale of 2 neighbours, and for 16 neighbours, with a total number of 500 particles. As can be seen, near the left endpoint, where the function does not change very much, the computed gradient is very scattered. Increasing the size of the smoothing length to include more particles within each summation causes the function and gradient values to be smoothed over a large range, this reduces the scattering in areas where the gradient does not change very fast. On the right side of the interval, the gradient changes very quickly over a small range. If the smoothing length is too large, the gradient is not represented very well. As can be seen in figure 4.4b, the particle approximation of the gradient on the right side of the graph is not well represented because its values are smoothed out over a range that is comparable to the range in which the gradient changes.

For this particular function, there is a critical value for the number of particles that should be in the smoothing length, but this is not universal, different functions will have a minimum at different locations.



(a) 2 particles in smoothing length



(b) 16 particles in smoothing length

Figure 4.4: Particle approximation of function f, with random noise.

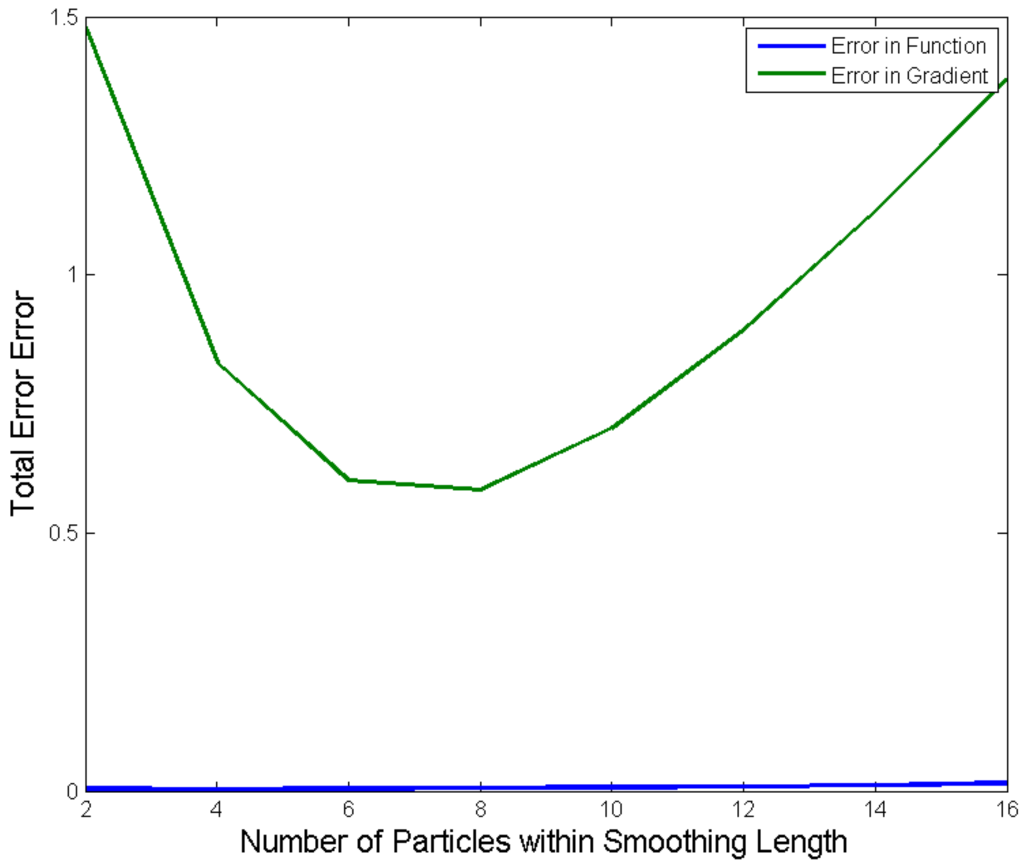


Figure 4.5: Error plot for approximating the derivative of the function $f(x)$ using the corrected particle gradient approximation with varoius number of particles and smoothing length.

Unlike the finite difference methods where the resolution of the simulation is determined by the grid spacing, the SPH approximation is determined by the size of the smoothing length, h. Even though the distance between particles, Δx, can be much smaller than the smoothing length, h, the system will not converge according to Δx. The smoothing length is responsible for smoothing out high frequency noise that could possibly develop in the system. By smoothing the values out over a distance of h, the resolution of the simulation can only be accurate up to an order of h^2, as was found in equation 2.11

4.3 Two-Dimensional Tests

To test some of the fluid simulation capabilities of SPH, we performed three two-dimensional experiments to test the various aspects of SPH. We tested the surface pressure condition for the fluid surface when solving the Pressure-Poisson Equation (Section 4.3.1), an imposed artificial viscosity for system stability (Section 4.3.2) and the type of kernel used (Section 4.3.3). The experiments involved looking at a square volume of fluid under the effect of surface tension. Surface tension will act on the square droplet and deform it into a circle. We found that the best results occurred if the surface particle pressure of the fluid was non-zero, artificial viscosity was turned on and the third-order polynomial kernel was used instead of the sixth-order polynomial. Table 4.2 shows the components that were varied for each experiment along with components that resulted in a stable simulation. The stable simulation outcome is shown in Figure 4.6. The parameters that were left constant in all three experiments are summarized in Table 4.3. The experiments were performed and compared with the stable case that produced the best results.

Unlike the traditional SPH method for compressible fluids where the mass of the fluid particle is fixed and the density is determined using Equation 2.66; in our tests the density of the particles is fixed, and the mass of each particle is determined by taking the total mass of the fluid (density $\times$ volume) and dividing it by the total number of particles in the system.

	Surface Pressure	Artificial Viscosity	Kernel
Stable	Non-Zero	Yes	W_3
Experiment 1	Zero	Yes	W_3
Experiment 2	Non-Zero	No	W_3
Experiment 3	Non-Zero	Yes	W_6

Table 4.2: Components of the SPH simulation that were varied in each experiment

4.3.1 Experiment 1: Surface Particle Pressure

When solving the Pressure-Poisson Equation, homogeneous Dirichlet boundary conditions are required on the fluid interface. In SPH, the boundary conditions are that the surface particles' pressure should be zero, or at constant atmospheric pressure, similar to the condition for fixed-grid methods. If this is done, all surface particles will have the same pressure. If any two surface particles come too close to each other due to the morphing of the fluid surface, there will be no pressure gradient to force the particles apart, this will cause clumping of particles. The clumping of the particles along with the surface tension algorithm will produce unphysical results and can cause the system to diverge. To solve this issue, the fluid interface is assumed to be one particle spacing away from the surface particles, see Figure 4.7, rather than on the surface particles. Surface particle, i and interior particle j are mirrored across the imaginary fluid interface and their pressure is negated. The negation of the pressure is to insure that the pressure is zero at the fluid interface.

A slight modification is required to be made to the SPH Laplacian. If particle i is

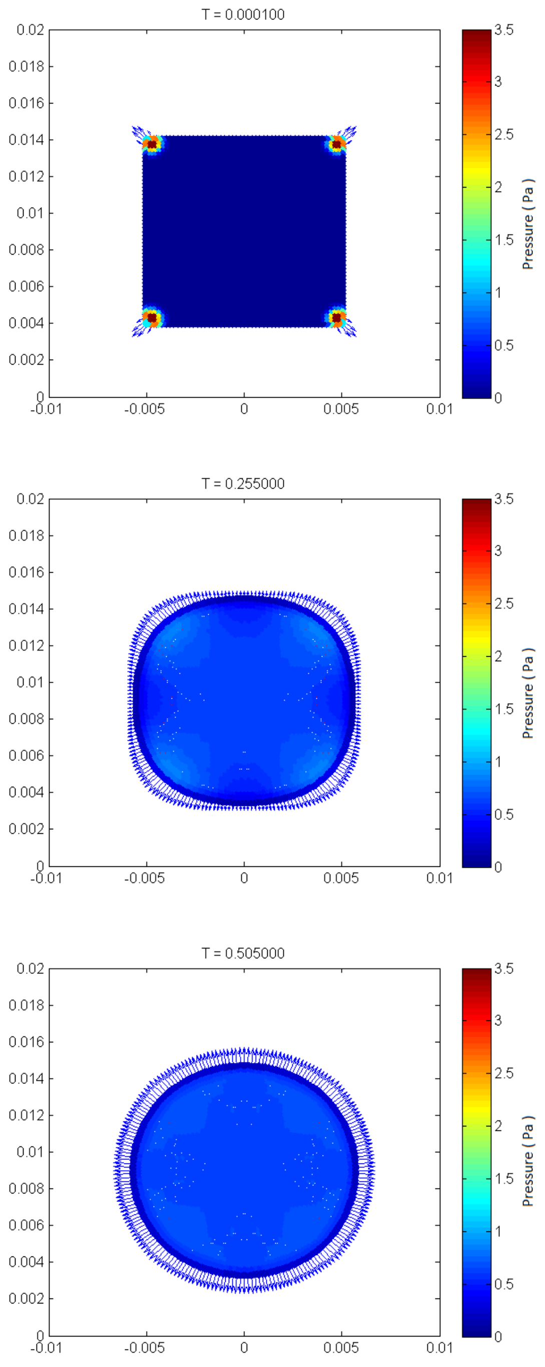


Figure 4.6: Evolution of a square droplet using non-zero surface pressure, artificial viscosity and third-order polynomial kernel.

Parameter	Value
Number of Particles N	2600
Density ρ_i	$1000\ \frac{kg}{m^3}$
Viscosity $[\mu_0, \mu_\infty]$	$[0.002, 0.004]$ Pa · s
Cross Model Parameters $[C, a]$	$[2, 2]$
Surface Tension σ	$7.2 \times 10^{-3}\ \frac{N}{m}$
Particle Spacing	0.0002 m
Smoothing Length h	0.0007 m
Side Length	0.01 m

Table 4.3: SPH parameters used in simulations.

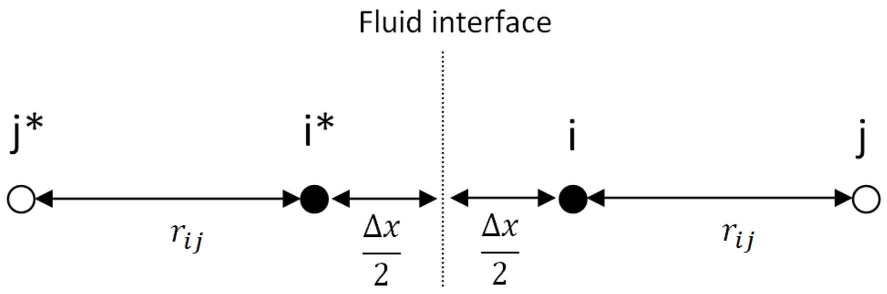


Figure 4.7: Real and mirrored (*) particles across the imaginary fluid interface.

a surface particle and particle j is an interior fluid particle, then the Laplacian of the pressure for particle i is written as

$$\nabla \cdot \left(\frac{\nabla P}{\rho}\right)_i = \sum_{j=1}^{N} \frac{8m_j}{(\rho_i + \rho_j)^2} \left(P_i - P_j\right) F\left(|\mathbf{r_{ij}}|\right) + \sum_{j=1}^{N} \frac{8m_j}{(\rho_i + \rho_j)^2} \left(P_i + P_j\right) F\left(|\mathbf{r_{ij}}| + \Delta x\right), \tag{4.8}$$

where

$$F(r) = \frac{\partial W}{\partial r}\frac{1}{r}, \tag{4.9}$$

otherwise, Equation 3.61 is used. When computing the gradient, using Equation 3.8, at the surface particles, the mirrored particles must also be included into the summation.

Figure 4.8 show the evolution of a droplet without particle surface pressure (Figure 4.6 shows the evolution with surface pressure). As can be seen, when the surface deforms under the affect of surface tension, particles are forced together. Since there is no pressure gradient between neighbouring surface particles, the surface tends to stay deformed and cause unphysical behaviour on the surface of the fluid. If Equation 4.8 is used, the surface particles of the fluid can be forced apart by the pressure gradient. This allows for much better physical results and the surface tension algorithm does not produce unrealistic results due to particles being too close to each other.

4.3.2 Experiment 2: Artificial Viscosity

It was found that using an artificial equation of state to approximate incompressibility was very unstable unless an artificial viscosity term was included [13]. The artificial viscosity term introduced by Monaghan is usually coupled with the pressure gradient term:

$$\left\langle \frac{\nabla P}{\rho} \right\rangle_i = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla W_{ij} \tag{4.10}$$

where

$$\Pi_{ij} = \begin{cases} \frac{-\alpha c_s \phi_{ij} + \beta \phi_{ij}^2}{\bar{\rho}_{ij}} & , \phi_{ij} < 0 \\ 0 & , \phi_{ij} > 0 \end{cases} \tag{4.11}$$

$$\phi_{ij} = \frac{h \mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{|\mathbf{r}|_{ij}^2 + 0.01h^2} \tag{4.12}$$

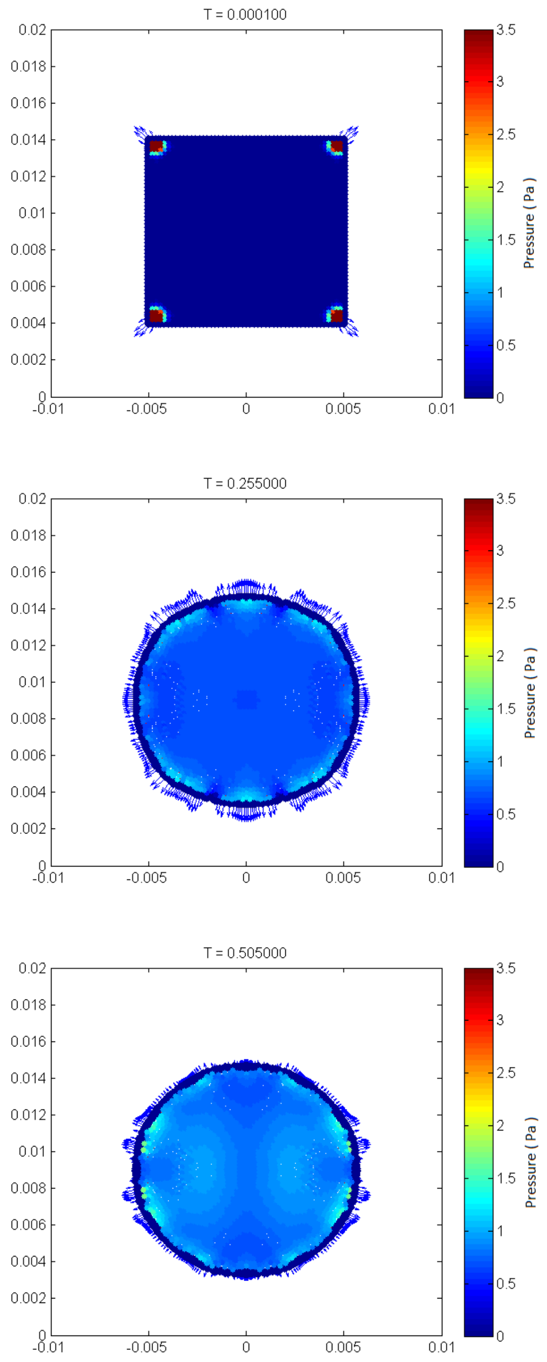


Figure 4.8: Evolution of a square droplet with zero surface pressure boundary condition.

The artificial viscosity has the effect of slowing particles down as they move towards each other. The term is symmetric in the indices i and j so the artificial viscosity conserves linear and angular momentum and is also zero for rigid body rotation. The constants c_s, α and β are the speed of sound and the linear and quadratic viscosity coefficients. The linear term is the most important for stability while the quadratic term is usually used in scenarios where fluid particles are approaching each other at high speeds such as supersonic flows. In most of the SPH literature, and in our tests, β was set to zero. The linear term, α is usually set according to the problem that is trying to be solved. The value of α is set to be the smallest value that keeps the system stable. The only other place the speed of sound appears is in the artificial equation of state (Equation 3.55). Since we are using the pressure correction method and assuming incompressibility we do not have an actual sound speed in the fluid, so the two constants α and c_s can be combined into one constant. We chose to keep them as separate constants to be consistent with the original formulation by Monaghan. Since h appears in the numerator of the artificial viscosity term, the effect of increasing the resolution of the simulation (increasing the number of particles and decreasing the smoothing length, h) has the effect of lowering the contribution of the artificial viscosity. In the limit as $h \to 0$ the artificial viscosity vanishes. Monaghan also showed that the extra viscosity that is added due to this term is approximately equal to $\alpha h c_s$. Many authors that work with the pressure correction method for SPH do not include the artificial viscosity term, but it was found that in our cases, the artificial viscosity was necessary to keep the system stable. In our tests, we take $\alpha = 10^{-6}$, $\beta = 0$ and $c_s = 120$.

Equation 4.13 is usually employed when an artificial equation of state is used to determine the pressure. If the pressure is determined using the Pressure Poisson Equation, then the artificial viscosity term in Equation 4.13 will prevent the pressure gradient from properly moving the particles into a divergence free velocity field. This

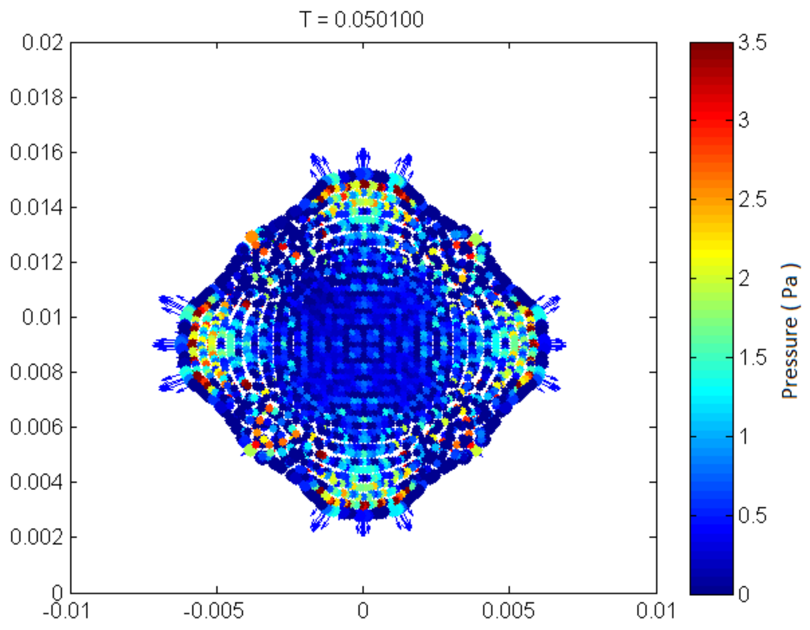


Figure 4.9: Simulation of the square droplet without artificial viscosity.

will eventually cause particles to move closer and closer together and therefore will not conserve volume. Instead of coupling the artificial viscosity with the pressure gradient as is normally done, we couple it with the divergence of the stress tensor.

$$\left\langle \frac{\nabla \cdot \boldsymbol{\tau}}{\rho} \right\rangle_i = \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\tau}_i}{\rho_i^2} + \frac{\boldsymbol{\tau}_j}{\rho_j^2} \right) \cdot \nabla W_{ij} + \sum_{j=1}^{N} m_j \Pi_{ij} \nabla W_{ij} \tag{4.13}$$

Figure 4.9 shows a square droplet without artificial viscosity. The simulation became unstable very quickly and diverged within 0.1 seconds of simulated time.

4.3.3 Experiment 3: Third-Order Polynomial Kernel for Pressure

Much of the literature uses a Gaussian-like kernel for most of the summations, and uses the third order polynomial (spikey kernel) for the pressure. The reason for this is the derivative of the Gaussian-like kernels (Equation 2.18) go to zero at $r = 0$. The third order polynomial on the other hand goes to infinity. This is crucial because the pressure gradient is responsible for keeping particles apart. If the derivative of the kernel goes to zero as the distance between particles decrease, then the pressure gradient will not be able to keep particles from approaching each other. This will cause particles to bunch together. Since the third order polynomial goes to infinity as particles approach each other, there will always be a greater and greater force keeping particles apart as they approach each other. Although one must be careful in the coding to make sure that during the summation in calculating the pressure gradient, Equation 3.8 does not include the index $j = i$. If this index is included then the distance between the particle and itself is zero making the gradient of the kernel singular. It is also important to use the third order polynomial kernel in the computation for the artificial viscosity for the same reason that the force keeping the particles apart should increase without bounds as two particles approach each other. In much of the literature, when the pressure correction method was employed to enforce incompressibility, the third order polynomial kernel is not used, rather a Gaussian-like kernel was used for all summations. In our tests, it was found that the third order polynomial should be used in constructing the Pressure-Poisson coefficient matrix as well as to compute the pressure gradient, Equation 4.13.

Figure 4.10 shows the evolution of the square droplet using the 6^{th} order Gaussian-like polynomial kernel (Equation 2.18) instead of the cubic polynomial (Equation 2.19). As can be seen in the figure, the particles start to bunch together due to the fact that the gradient of the kernel goes to zero as the distance between particles decreases.

4.4 Convergence Rate of Conjugate Gradient Method for Solving the Pressure-Poisson Equation

In the Pressure Poisson Equation given by Equation 3.61 the derivative of the smoothing function appears on both sides of the equation. Since the derivative appears on both sides, so does the scalar multiplier that makes the integral of the kernel equal to one (table 2.1). Since the coefficient is the same for all particles, it can be removed from the system of equations which corresponds to a diagonal preconditioning. Aside from removing the scalar coefficient from the kernel, it was found that writing the kernel in the following way also increased the convergence rate of the algorithm.

$$W_3^*(r) = (h-r)^3 \tag{4.14}$$

$$\frac{\partial W_3^*}{\partial r}\frac{1}{r} = -3\frac{(h-r)^2}{r} \tag{4.15}$$

Using Equation 4.14 for the kernel (note the scalar coefficient was removed) drastically increased the convergence rate of the conjugate gradient algorithm. The convergence rate of the conjugate gradient algorithm increased to the point where it only required one iteration to converge to a solution with a residual norm of 10^{-10}, rather than upwards of 30 iterations if Equation 2.19 was used. The conjugate gradient algorithm requires an initial guess for the solution to the system of equations. By using the previous iteration's values for pressure as the initial guess and knowing that the pressure shouldn't change very much between iterations if the flow is fairly smooth a convergence within one iteration is justified.

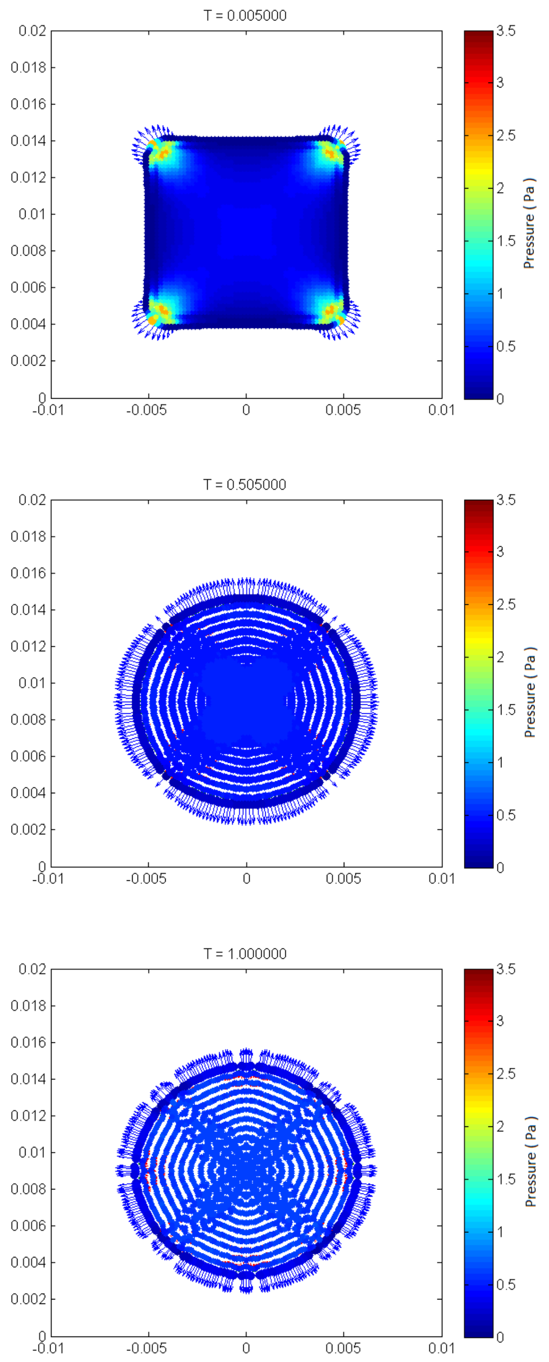


Figure 4.10: Evolution of a square droplet using a Gaussian-like kernel.

Chapter 5

Discussion and Conclusion

The SPH implementation in this thesis was done in two dimensions for simplicity in determining the capabilities of SPH. An extension to three dimensions is fairly non-trivial when computing the SPH summations. The surface tension algorithm, in three dimensions, increases significantly in mathematical and computational complexity. Determining surface particles in three-dimensions is significantly more involved than in two-dimensions. Haque and Dilts provide a three-dimensional algorithm to find surface particles as well [6]. This method involves looking at intersections of spheres rather than circles. Consider a particle, i, and neighbouring particles j. When the two spheres of these particles intersect, they produce a circle of intersection. The circle of intersection lies on the surface of sphere i. All circles of intersections are determined by the intersection of nearby spheres and each of these circles will lie on particle i. The circles of intersections that now lie on sphere i are checked with every other circle to determine their coverage similar to the method for two-dimensions. If every circle on sphere i is fully covered, then particle i is an interior particle, otherwise it is a surface particle. The circles that lie on the surface of the sphere do not lie on the same plane, so a the normal direction of the circle must also be taken into account to determine the coverage of the circles. Haque and Dilts have provided a mathematical formulation for determining the coverage of a circle in three dimensions. Once the surface particles have been found, the surface reconstruction is performed in a similar fashion except using the following basis functions: $1, x, y, xy, x^2, y^2$.

SPH first originated in the astrophysical community to simulate the gas dynamics in galaxies. The method is more suited for applications in that area rather than in the liquid domain. The appeal to use SPH in astrophysics is due to the fact that SPH works better when simulating compressible fluids, since a non-stiff equation of state is defined and the a Pressure-Poisson equation does not need to be solved. Also, due to the fact that galaxy simulations usually employ open domains, this eliminates the need for strict boundary conditions.

Unfortunately there are many aspects of SPH that does not make it a good choice of technique to use for simulating droplet impacts. First, the surface tension algorithm, although can detect surface particles quite sufficiently, has a difficult time conserving momentum. There is no method in place to guarantee that the total flux of the the surface tension force around the fluid surface is zero. If this surface flux is non-zero, it will cause extra energy to be inputted into the system and the droplet will eventually start to accelerate away from its true position. In our tests, momentum was conserved for a square droplet deforming into a circle, but was not conserved for an oscillating ellipse. One method that could possibly be used to ensure that the surface flux is zero is to divide the total sum of the forces by the number of particles with surface tension and then subtract that quantity from all the surface particles. This will ensure that the sum is always zero. This was tested but did not produce expected results. Figure 5.1 shows the evolution of an ellipse using the previously stated method. As can be seen in the 4^{th} frame, the left side of the ellipse is not symmetric with the right side of the ellipse. Deeper investigation would need to be conducted to determine a way to conserve momentum with external surface tension.

Solving of the Pressure-Poisson equation is very time consuming, even with the

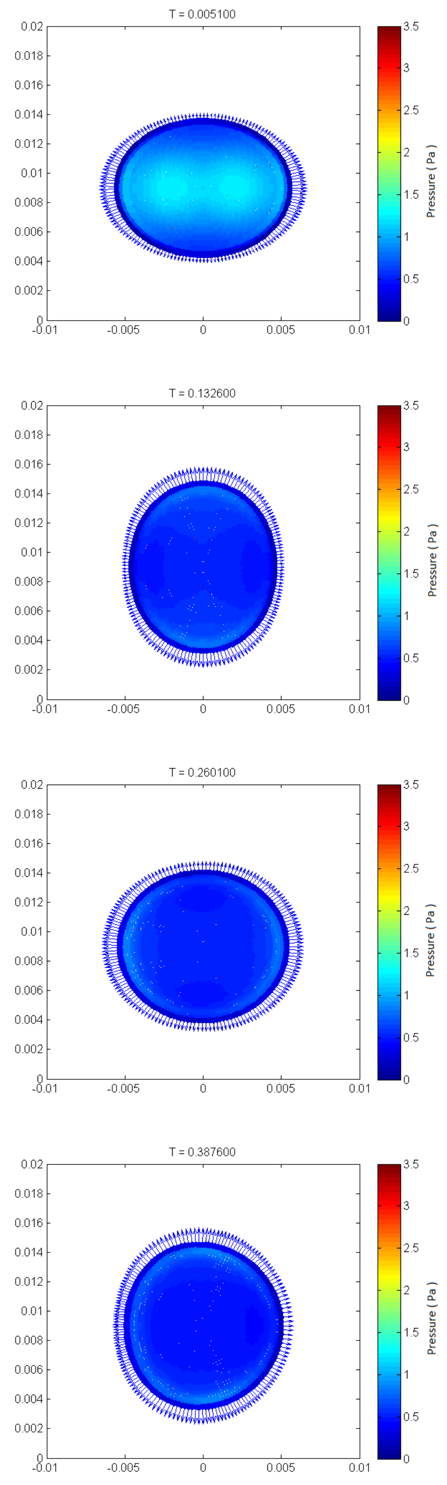


Figure 5.1: Evolution of an elliptical droplet with average surface tension flux removed.

fast convergence rate of the conjugate gradient algorithm. Almost 60% of the time required to solve the Pressure-Poisson equation goes towards the construction of the matrix. Since the particles are not static, the coefficient matrix changes in each time step so it must be reconstructed during each iteration. This cannot be avoided unless one uses a fixed grid method. This drastically increases the computation time compared to fixed grid methods.

The second derivative approximation is not very accurate, even when using the finite difference approximation combined with the SPH first derivative, Equation 2.69. Most of the error occurs on the edges of the fluid due to there not being as many particles within the support domain of the kernel. For incompressible fluids, the pressure force is entirely responsible for the fluid flow to stay divergence free. Since the Pressure-Poisson Equation is constructed using the Laplacian, this can lead to unpredictable behaviour due to this error.

Although Smoothed Particle Hydrodynamics has its advantages and disadvantages, the advantages do not fall in favour with the models that need to be solved for non-Newtonian droplet impact. The reasons for this as outlined above are the difficulty in solving the Pressure-Poisson equation on the surfaces as well as on solid boundaries, the difficulty in conserving momentum with the surface tension and the speed of the computation. The time required to run the simulations presented in this thesis took on the order of a few hours. By extended this method to three dimensions, the computation time would increases significantly. The best solution to solving this problem might be to use an alternative method such as finite volume or finite element methods.

Bibliography

[1] HA Barnes, JF Hutton, and K. Walters. An introduction to rheology, rheology series 3. *An Introduction to Rheology: Rheology Series*, 3, 1989.

[2] G.K. Batchelor. *An introduction to fluid dynamics*. Cambridge Univ Pr, 2000.

[3] S.J. Cummins and M. Rudman. An SPH projection method. *Journal of computational physics*, 152(2):584–607, 1999.

[4] G.A. Dilts. Moving least-squares particle hydrodynamics II: conservation and boundaries. *International Journal for Numerical Methods in Engineering*, 48(10):1503–1524, 2000.

[5] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics-theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[6] A. Haque and G.A. Dilts. Three-dimensional boundary detection for particle methods. *Journal of Computational Physics*, 226(2):1710–1730, 2007.

[7] L. Hernquist and N. Katz. Treesph-a unification of sph with the hierarchical tree method. *The Astrophysical Journal Supplement Series*, 70:419–446, 1989.

[8] S.M. Hosseini, M.T. Manzari, and S.K. Hannani. A fully explicit three-step SPH algorithm for simulation of non-Newtonian fluid flow. *International Journal of Numerical Methods for Heat & Fluid Flow*, 17(7):715–735, 2007.

[9] S.H. James. *Scientific and legal applications of bloodstain pattern interpretation*. CRC, 1998.

[10] G.R. Liu and MB Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific Pub Co Inc, 2003.

[11] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977.

[12] J Monaghan. An introduction to SPH. *Computer Physics Communications*, 48(1):89–96, January 1988.

[13] J.J. Monaghan. Simulating free surface flows with SPH. *Journal of computational physics*, 110:399–399, 1994.

[14] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33(3):333–353, 2000.

[15] K.W. Morton and D.F. Mayers. *Numerical solution of partial differential equations: an introduction*. Cambridge Univ Pr, 2005.

[16] Y. Saad and Y. Saad. *Iterative methods for sparse linear systems*. PWS Pub. Co., 1996.

[17] S. Shao and E.Y.M. Lo. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, 26(7):787–800, 2003.

[18] Alexandre Tartakovsky and Paul Meakin. Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E*, 72(2):1–9, August 2005.

[19] M. Zhang. Simulation of surface tension in 2D and 3D with smoothed particle hydrodynamics method. *Journal of Computational Physics*, 229(19):7238–7259, 2010.

[20] MY Zhang, H. Zhang, and LL Zheng. Simulation of droplet spreading, splashing and solidification using smoothed particle hydrodynamics method. *International Journal of Heat and Mass Transfer*, 51(13-14):3410–3419, 2008.