Applying Contact Angle to a Two-Dimensional Smoothed Particle Hydrodynamics (SPH) model on a Graphics Processing Unit (GPU) Platform

Amirsaman Farrokhpanah
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2012

Abstract

A parallel GPU compatible Lagrangian mesh free particle solver for multiphase fluid flow based on SPH scheme is developed and used to capture the interface evolution during droplet impact. Surface tension is modeled employing the multiphase scheme of Hu et al. [1]. In order to precisely simulate the wetting phenomena, a method based on the work of Šikalo et al. [2] is jointly used with the model proposed by Afkhami et al. [3] to ensure accurate dynamic contact angle calculations. Accurate predictions were obtained for droplet contact angle during spreading.

A two-dimensional analytical model is developed as an expansion to the work of Chandra et al. [4]. Results obtain from the solver agrees well to this analytical results.

Effects of memory management techniques along with a variety of task assigning algorithms on GPU are studied. GPU speedups of up to 120 times faster than a single processor CPU were obtained.
Acknowledgments

First and foremost I offer my sincerest gratitude to my supervisor, Professor Javad Mostaghimi, for his constant support and for generously offering advice and encouragement.

I am grateful to my colleague and friend, Dr. Babak Samareh for helping me along the way and for his time and valuable discussions.

I would like to thank everyone in the CACT lab that either directly or indirectly played part in this work.

Finally, I thank my parents for supporting me throughout all my studies.
To my parents,
Mitra and Mehdi
Table of Contents

Acknowledgments ........................................................................................................................ iii
Table of Contents .......................................................................................................................... v
List of Tables .............................................................................................................................. viii
List of Figures ............................................................................................................................. viii
1 Introduction ................................................................................................................................ 1
  1.1. overview .............................................................................................................................. 1
2 Smoothed Particle Hydrodynamics .......................................................................................... 4
  2.1. principles ............................................................................................................................. 4
    2.1.1. integral representation of a function ....................................................................... 4
    2.1.2. integral representation of derivative of a function .................................................. 6
    2.1.3. particle approximation using riemann sums ........................................................... 7
  2.2. conservation of mass and momentum ................................................................................. 9
    2.2.1. Navier-Stokes equations in lagrangian form ........................................................... 9
    2.2.2. SPH representation of continuity equation ........................................................... 11
    2.2.3. SPH representation of momentum equation ........................................................... 13
  2.3. equation of state ................................................................................................................ 15
  2.4. smoothing functions .......................................................................................................... 17
  2.5. boundary treatments .......................................................................................................... 19
  2.6. time integration ................................................................................................................. 23
3 Drop Formation ....................................................................................................................... 25
  3.1. drops in single-phase SPH ................................................................................................ 25
  3.2. drops in multi-phase SPH ................................................................................................. 28
    3.2.1. interface tracking .................................................................................................. 28
    3.2.2. continuum surface force (CSF) ............................................................................. 29
3.2.3. SPH implementation ................................................................. 30

4 SPH Droplet Spread and Impact ......................................................... 34

4.1. contact angle in SPH .............................................................. 34

4.2. validation ................................................................................. 40

4.2.1. oscillating rod test .............................................................. 40

4.2.2. stationary drop with contact angle of 90° ......................... 41

4.2.3. drops in equilibrium ........................................................... 46

4.2.4. convergence test ................................................................. 50

5 Case Study: 2-D Droplet Impact ......................................................... 53

5.1. analytical 2-D cartesian model .............................................. 53

5.2. 2-D drop impact with constant contact angles .................... 58

6 SPH on CUDA ................................................................................ 64

6.1. GPU architecture ................................................................. 64

6.2. SPH implementation on GPU ................................................. 66

6.3. searching for neighboring particles ....................................... 67

6.4. dimension of kernels ............................................................... 68

6.4.1. one particle per multiple threads ...................................... 69

6.4.2. one particle per thread ....................................................... 76

6.5. memory management .............................................................. 77

7 Closure ......................................................................................... 79

7.1. concluding remarks ............................................................... 79

7.2. future work ......................................................................... 79

References ......................................................................................... 81
List of Tables

Table 1 Comparison of measured values of spread factor with predictions, showing results from axisymmetric model of pasandideh-fard et al. and 2-D cartesian results obtained using assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93) ................. 56
List of Figures

Figure 2.1 A sample smoothing function being defined around the point $x_0$ on a one-dimensional domain.......................... 6

Figure 2.2 Integral approximation using riemann sums, more divisions made to the domain would result in more accurate results. $f(x_i)$ can be approximated either with the value of $f(x)$ at the middle of each division or any other arbitrary points......................................................... 7

Figure 2.3 A smoothing function acting on a two-dimensional domain. dots represent particles each with a volume of $\delta V$ sweeping the whole surface.............................................................. 8

Figure 2.4 Calculation of ghost particle velocities based on real particle velocities for constructing a no-slip boundary condition............................................................... 21

Figure 2.5 a) when fluid particle only has fluid neighbors ($\chi_i = 1$) b) when fluid particle has both fluid and wall neighbors ($0.5 \leq \chi_i \leq 1$) c) when fluid particle has equal fluid and wall neighbors ($\chi_i = 0.5$)........................................................................................................................................ 22

Figure 4.1 Quarter drop center of mass position versus time .......................................................... 40

Figure 4.2 Initial positioning of particles inside the drop. the second phase filling the domain outside the drop has not been shown. 833 particles are sweeping the surface of the drop......... 41

Figure 4.3 Contact angle deviations from $90^\circ$ for a half circle drop left to reach its equilibrium using three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = \alpha_{sl} = 1.0$.......................................................... 42

Figure 4.4 Contact angle deviations from $90^\circ$ for a half circle drop left to reach its equilibrium using one surface tension coefficient of $\alpha_{lg} = 1.0$ along with unit normal and $\nabla C$ correction ... 42

Figure 4.5 Total kinetic energy of all particles located inside quarter of the drop, using three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = \alpha_{sl} = 1.0$.......................................................... 43

Figure 4.6 Total kinetic energy of all particles located inside quarter of the drop, using one surface tension coefficient of $\alpha_{lg} = 1.0$ along with unit normal and $\nabla C$ correction ............... 44
Figure 4.7 Variations of average shear rate along the solid boundary, starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above), using three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = \alpha_{sl} = 1.0$................................. 45

Figure 4.8 Variations of average shear rate along the solid boundary, starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above), using one surface tension coefficient of $\alpha_{lg} = 1.0$ along with unit normal and $\nabla C$ correction ...................................................................................................................................... 45

Figure 4.9 Maximum deviation of unit normal vectors near the interface and away from the triple point. the black line with unfilled circles shows the case with three surface coefficients while the red line with unfilled triangles is related to the case of one surface tension coefficient with correction methods................................................................................................................ 46

Figure 4.10 Contact angle deviations from 60° when a no slip boundary condition is imposed. the red dashed line shows the case with three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = 1.0$ and $\alpha_{sl} = 0.5$. the black solid line is showing results for the case with $\alpha_{lg} = 1.0$ and normal and $\nabla C$ corrections.................................................................................................................................... 47

Figure 4.11 Variations of average shear rate at initial stages of drop’s evolution (averaged near time=0.25) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). the red dashed line shows the case with three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = 1.0$ and $\alpha_{sl} = 0.5$. the black solid line is showing results for the case with $\alpha_{lg} = 1.0$ and normal and $\nabla C$ corrections.................................................................................................................................... 48

Figure 4.12 Variations of average shear rate at equilibrium (averaged near time=4.4) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). the red dashed line shows the case with three surface tension coefficients of $\alpha_{lg} = \alpha_{sg} = 1.0$ and $\alpha_{sl} = 0.5$. the black solid line is showing results for the case with $\alpha_{lg} = 1.0$ and normal and $\nabla C$ corrections. 48

Figure 4.13 Spread factor of the drop (instantaneous diameter of drop divided by initial drop diameter). the green line with unfilled triangles is showing results for the case with $\alpha_{lg} = 1.0$
and normal and $\nabla C$ corrections where free slip condition is imposed on the boundary. The blue line with unfilled circles is also related to the same case with the difference of having a no slip boundary condition. The dashed red line demonstrates results of the three phase case with $\alpha_{tg} = \alpha_{sg} = 1.0$ and $\alpha_{st} = 0$ where a free slip boundary condition is imposed. The black solid line is also related to the same case with the difference of having a no slip boundary condition.

Figure 4.14 Contact angle deviations from $60^\circ$ when a free slip boundary condition is imposed. The red dashed line shows the case with three surface tension coefficients of $\alpha_{tg} = \alpha_{sg} = 1.0$ and $\alpha_{st} = 0.5$. The black solid line is showing results for the case with $\alpha_{tg} = 1.0$ and normal and $\nabla C$ corrections.

Figure 4.15 Variations of average shear rate at equilibrium (averaged near time=4.4) along the solid boundary with a free slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). The red dashed line shows the case with three surface tension coefficients of $\alpha_{tg} = \alpha_{sg} = 1.0$ and $\alpha_{st} = 0.5$. The black solid line is showing results for the case with $\alpha_{tg} = 1.0$ and normal and $\nabla C$ corrections.

Figure 4.16 Variations of average shear rate at equilibrium (averaged near time=4.4) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). The red line with unfilled triangles shows the case with the resolution of 0.5/105. The green solid line is related to the resolution of 0.5/85. The blue solid line shows results of the 0.5/65 case while the black line with unfilled circles shows the 0.5/45 case. In all cases, $\alpha_{tg} = 1.0$ and normal and $\nabla C$ corrections are used.

Figure 4.17 Total kinetic energy of all particles located inside the quarter of drop, using surface tension coefficient of $\alpha_{tg} = 1.0$ along with unit normal and $\nabla C$ corrections. Plotted solid lines with colors of red, green, blue, and black represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively.
Figure 4.18 Spread factor of the drop (instantaneous diameter of drop divided by initial drop diameter) for the case with $\alpha_{tg} = 1.0$ and normal and $\nabla C$ corrections. Plotted data with red line with unfilled triangles, green solid line, blue solid line, and black line with unfilled circles represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively................. 52

Figure 4.19 Contact angle deviations from 60° with a no slip boundary condition for various resolutions. Plotted solid lines with colors of red, green, blue, and black represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively..................................................... 52

Figure 5.1 2-D cartesian drop impacting on a surface and generating a thin film of liquid beneath ....................................................................................................................................................... 53

Figure 5.2 Impacting drop on a solid surface shown at initial rest position before the start of impact........................................................................................................................................... 59

Figure 5.3 Non-dimensional diameter ($D/D_0$) of spreading drops during impact versus non-dimensional time ($4\mu t/\rho D_0^2$) for various constant contact angles......................................................... 61

Figure 5.4 Impacted drops shown at their maximum expanded diameter for various constant contact angles imposed during impact.......................................................................................................................... 60

Figure 5.5 Maximum non-dimensional spread diameter for various contact angles, obtained from different models: axisymmetric model of Pasandideh-Fard et al., 2-D Cartesian results obtained from assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93). ....... 63

Figure 6.1 A one dimensional GPU grid holding three GPU blocks. Each block contains, as shown in the first one, contains of a number of threads (arranged in two dimensions here). ...... 65

Figure 6.2 Flowchart demonstrating SPH-GPU solver................................................................. 66

Figure 6.3 Runtime comparison on cpu and GPU for different neighbor search algorithms. In these tests, 2025 particles filled the domain while multiphase SPH formulations as discussed before were solved....................................................................................................................... 68
**Figure 6.4** Showing how each processing thread reads (or calculates) the value of smoothing function, then has to add the needed effects on the density using an atomic operation in order to avoid conflict with other threads.................................................. 70

**Figure 6.5** Simple reduction algorithm for the case of having large arrays................................. 73

**Figure 6.6** Improved reduction algorithm proposed by harris [61].............................................. 74

**Figure 6.7** Shows how reduction algorithms along with atomic operations can reduce the race condition complexity of a problem........................................................................................................ 75

**Figure 6.8** Runtime improvement by compiling more routines on GPU. in these tests, 2025 particles filled the domain while multiphase SPH formulations as discussed before were solved. ....................................................................................................................................................... 76

**Figure 6.9** GPU performance analysis versus the number of particles in each subdomain. (○) atomic operations used and most variables calculated once and saved for referencing, (Δ) no atomic operations used and variables recalculated on demand, and (□) atomic operations used along with reduction algorithms and variables recalculated on demand ........................................ 78
Chapter 1
Introduction

1.1. Overview

Smooth particle hydrodynamics or SPH was introduced and developed by Gingold and Monaghan [5] and Lucy [6] in 1977. In SPH, computational domain is discretized using fluid particles. Each particle has density and mass to represent a lump of fluid moving around with the velocity of the fluid at that location in a Lagrangian manner. Properties of these particles are smoothed over a distance known as the smoothing length. This means that the properties of a particle of interest can be calculated from its neighboring particles. The contribution of neighbors is weighted using a kernel function which mostly depends on the distance of neighboring particles.

Since 1977, SPH has been excessively used in simulating different physical phenomena in fields like astrophysics, fluid sciences, oceanography, ballistics, etc. One of the major subjects studied in SPH is interfacial flows. Practical studies like tsunami simulations [7], simulation of floating bodies like ships [8], and multiphase studies [1, 9, 10] are among them.

1.2. Advantages of SPH

There are many advantages associated with SPH that make it a good candidate for simulations. One of the major ones is the mesh free nature of SPH. Being Lagrangian and mesh free omits all the problems related to meshing complex geometries. Fluid particles can be initially arranged evenly inside the desired geometry. By starting the simulations, density variations and pressure forces would force particles to move around till they reach a uniform density distribution.

In addition, in many forms of SPH formulation, mass is conserved exactly. As long as particles are not added or lost in the domain, the mass that is carried by each particle is conserved by definition. This makes SPH even a very good candidate for being coupled with some mesh methods like level set that might be open to mass loss.
Moreover, pressure field in SPH can be calculated from equation of state rather than solving linear systems of equations. Although there are SPH studies like Hu et al. [11] that use incompressible solvers like the Poisson's equation, it has been shown [12] that by utilizing the equation of state, incompressible problems can be solved in a reliable and much faster manner. This makes SPH suitable for application in real-time simulations, such as animation and gaming industries.

1.3. Surface tension in SPH

Many multiphase studies have been focused on droplet formation and impact in applications such as spray coating, spray cooling, and inkjet printing. Various investigations have been conducted using analytical, experimental, and numerical methods.

For numerical simulation of droplets, a reliable surface tension model is needed. In SPH, droplets have been modeled using different methods. One of these methods is by using microscopic forces between particles like the model of Nugent et al. [12] which is based on the Van der Waals equation. Some studies have used this method for simulating drop impact situations like in works like Zhang et al. [13] and Xiong et al. [14]. Although the reported results are satisfying, the surface tension model used in these studies needs calibration and is resolution dependent. These studies also lacked consideration of contact angle variations at triple point.

To overcome this, other studies have been focused on adding surface tension effects to SPH using macroscopic schemes. In these schemes, surface tension has been modeled by means of macroscopic surface tension forces like the method suggested by Hu et al. [1] based on the CSF method of Brackbill et al. [15]. These methods have been shown to be resolution independent.

Das et al. also studied different methods of imposing contact angle to a sessile and inclined droplet using particle rearrangement [16, 17]. Hu et al. [1] presented a model in which contact angle for sessile droplet was calculated indirectly by the means of added surface tension forces in accordance to Young-Laplace equation. Reported results of these methods are satisfying although there is still room for improving speed and accuracy.
1.4. Objectives

In the present study, the main focus is on improving the accuracy of the obtained contact angle at the triple point. For this purpose, the model of Hu et al. [1] in two dimensions is used in combination with the methods previously utilized in Volume of Fluid Method (VOF) in the works of Šikalo et al. [2] and Afkhami et al. [3] in order to better capture the dynamic contact line behaviors.

Results are then compared against an analytical model which is an expansion to the study of Chandra et al. [4]. The two dimensional model is optimized to be executed on Graphic Processing Units (GPU).

1.5. This thesis

The remainder of this thesis is arranged as follows. An introduction to SPH principals and formulations along with appropriate numerical treatments are reviewed in Chapter 2. Chapter 3 reviews available multi-phase and single-phase models in formation of SPH drops. Chapter 4 covers the proposed method for implementing contact angle on SPH drops. In chapter 5, a case study related to drop impact is conducted and compared against the developed analytical model. Chapter 6 covers the methods and algorithms used in making the solver available and efficient for execution on the Graphic Processing Units.
Chapter 2
Smoothed Particle Hydrodynamics

2.1. Principles

In the following chapter, some basic concepts and formulations of Smoothed Particle Hydrodynamics (SPH) have been studied. The fundamental topics mentioned in this chapter are from a single phase point of view. In next chapters, multiphase implementation of SPH which was actually used in this study would be introduced in more details.

2.1.1. Integral representation of a function

Considering the definition of the Dirac delta ($\delta$) function in the form of

$$\delta(x - x_0) = \begin{cases} \infty, & x = x_0 \\ 0, & x \neq x_0 \end{cases}$$

would lead to the following identity

$$\int_{-\infty}^{\infty} \delta(x - x_0) dx = 1$$

For the integral representation of a function, $f(x)$, it can easily be shown that

$$\int_{\phi} f(x) \delta(x - x_0) dx = f(x_0)$$

with $\phi$ being part of the volume domain which contains $x_0$ and $f(x)$ is defined and continuous on it. A more general case of this integral interpolation can be obtained by substituting delta function with a smoothing function like $W(x - x_0, h)$ where

$$\lim_{h \to 0} W(x - x_0, h) = \delta(x - x_0)$$

and $W$ is normalized so
\[
\int_{-\infty}^{\infty} W(x - x_0, h) \, dx = 1 \quad (5)
\]

\( h \) which is usually accompanied by an integer multiplicand, \( k \), determines the radius from \( x_0 \) on which the smoothing function is non-zero (as shown in figure 2.1). Therefore equation (3) gets the form of

\[
\int_{\phi} f(x) \, W(x - x_0, h) \, dx \equiv f(x_0) \quad (6)
\]

which is only an approximation of \( f(x_0) \) unless \( W(x - x_0, h) \) is the Dirac delta function [18]. A schematic view of \( W(x - x_0, h) \) on a one-dimensional domain is shown in figure 2.1. Note that \( x \) and \( x_0 \) can be vectors of position in a one, two, or three dimensional domain and \( dx \) consequently would be an infinitesimal element of line, surface, or volume, respectively. Another condition enforced on the smoothing function is being compact

\[
W(x - x_0, h) = 0 \quad \text{if} \quad |x - x_0| \geq kh \quad (7)
\]

where \( kh \) defines the support domain \( \phi \) of the smoothing function. Having a compact condition would lead to localized integrations only on the support domain of the smoothing function. In other words, the kernel approximation would act only inside the domain of \( \phi \) and remains zero outside this domain. Depending on whether each smoothing function is defined on a one, two or three dimensional domain, different types of functions can be defined which have to possess all three conditions previously defined here. Smoothing functions will be discussed later in more detail.

To investigate equation (6) in more details, the Taylor series expansion of \( f(x) \) around \( x_0 \) gives

\[
\int_{\phi} f(x) W(x - x_0, h) \, dx = \int_{\phi} \left[ f(x_0) - f'(x_0)(x - x_0) + r((x - x_0)^2) \right] W(x - x_0, h) \, dx
\]

\[
= f(x_0) \int_{\phi} W(x - x_0, h) \, dx - f'(x_0) \int_{\phi} [(x - x_0)] W(x - x_0, h) \, dx
\]

\[
+ \int_{\phi} [r((x - x_0)^2)] W(x - x_0, h) \, dx
\]
By definition, the identity \( \int_{\phi} W(x - x_0, h) \, dx = 1 \). By choosing \( W(x - x_0, h) \) to be an even function of \( x - x_0 \), we can assume \( \int_{\phi} [(x - x_0)] W(x - x_0, h) \, dx = 0 \). Therefore

\[
\int_{\phi} f(x) W(x - x_0, h) \, dx = f(x_0) + \int_{\phi} [r((x - x_0)^2)] W(x - x_0, h) \, dx
\]

(8)

Figure 2.1. a sample smoothing function being defined around the point \( x_0 \) on a one-dimensional domain.

2.1.2. Integral representation of divergence of a vector field

To approximate the spatial derivative of a vector field, \( f(x) \) in equation (6) can be replaced by \( \nabla \cdot f(x) \), giving

\[
\nabla \cdot f(x_0) \equiv \int_{\phi} \nabla \cdot f(x) \, W(x - x_0, h) \, dx
\]

(9)

Right hand side of equation (9) can be expanded to

\[
\nabla \cdot f(x_0) \equiv \int_{\phi} \nabla \cdot \left[ f(x) \, W(x - x_0, h) \right] dx - \int_{\phi} f(x) \cdot \nabla W(x - x_0, h) \, dx
\]

The use of divergence theorem on the first integral on the right hand side would lead to

\[
\nabla \cdot f(x_0) \equiv \int_{S} f(x) \, W(x - x_0, h) \cdot \vec{n} dS - \int_{\phi} f(x) \cdot \nabla W(x - x_0, h) \, dx
\]
The compact condition defined in equation (7) forces the first integral on the right hand side to be zero since \( W(x - x_0, h) \equiv 0 \) outside and on the surface of the domain \( \phi \) (figure 2.3). Hence

\[
\nabla \cdot f(x_0) \equiv - \int f(x) \cdot \nabla W(x - x_0, h) dx
\]

Equation (10) represents one of the main advantages of using this method of approximation. As it can be seen, divergence of a function (\( \nabla \cdot f(x_0) \)) can be calculated from the divergence of the smoothing function (\( \nabla W(x - x_0, h) \)). Therefore, knowing the function itself is enough for divergence calculation in equation (10).

2.1.3. Particle approximation using Riemann sums

Every definite integral of a function which is defined on its domain can be approximated using Riemann sums. For instance, an integral in the form of \( \int_a^b f(x) dx \) (figure 2.2.a) can be approximated by \( \sum_{i=1}^{n} f(x_i) \Delta x \) where \( \Delta x = (b - a)/n \). The value \( n \) here can be assumed as the number of divisions made to the domain. The approximation converges to the integral when \( n \) becomes infinitely large (figure 2.2.b and c). On each division, the value of the integrand is calculated based on a point located at that specific piece, \( f(x_i) \).

Equation (10) can also be approximated using the same method:

![Figure 2.2. Integral approximation using Riemann sums, more divisions made to the domain would result in more accurate results. \( f(x_i) \) can be approximated either with the value of \( f(x) \) at the middle of each division or any other arbitrary points.](image-url)
\[ \nabla \cdot f(x_0) \approx -\int_{0}^{x_0} f(x) \cdot \nabla W(x - x_0, h) dx \]

\[ \approx - \sum_{j=1}^{N} f(x_j) \cdot \nabla W(x_j - x_0, h) \Delta V_j \]  

(11)

Here the domain is divided into \( N \) equal pieces with the size of \( \Delta V_j \). In other words, the infinitesimal volume of \( dx \) is substituted by the finite volume of \( \Delta V_j \). Each piece can be thought of as a particle (dots in figure 2.3). If the finite volume of \( \Delta V_j \) is considered as the volume each particle occupies, a hypothetical mass can be calculated for each particle in the form of

\[ m_j = \rho_j \Delta V_j \]  

(12)

Figure 2.3. A smoothing function acting on a two-dimensional domain. Dots represent particles each with a volume of \( \Delta V \) sweeping the whole surface.

By substituting equation (12) into equation (11) would lead to

\[ \nabla \cdot f(x_0) \approx - \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \cdot \nabla W(x_j - x_0, h) \]  

(13)
Equation (13) shows that the value of spatial divergence of a function at a point like \( x_0 \) is proportional to sum of gradients of the smoothing function calculated on neighboring particles of \( x_0 \) as shown in figure 2.3.

As it can be seen from equations (6) and (9), the spatial gradient in equation (13), \( \nabla \), is taken with respect to \( x_j \). If a new spatial gradient, \( \nabla^* \), is defined with respect to \( x_0 \),

\[
\nabla W(x_j - x_0) = \frac{\partial W(x_j - x_0)}{\partial x_j} = \frac{\partial W(x_j - x_0)}{\partial (x_j - x_0)} \frac{\partial (x_j - x_0)}{\partial x_j} = \frac{\partial W(x_j - x_0)}{\partial (x_j - x_0)} = \\
= - \frac{\partial W(x_j - x_0)}{\partial (x_j - x_0)} \frac{\partial (x_j - x_0)}{\partial x_0} = - \frac{\partial W(x_j - x_0)}{\partial x_0} = - \nabla^* W(x_j - x_0) \quad (14)
\]

Therefore equation (13) can be rewritten in the form of

\[
\nabla \cdot f(x_0) \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \cdot \nabla^* W(x_j - x_0, h) \quad (15)
\]

### 2.2. Conservation of mass and momentum

In the previous sections, principles of SPH particle approximation were discussed. Here, conservation equations of mass and momentum will be reviewed in their Lagrangian form followed by their SPH approximations. The resulting discretized Navier-Stokes equations can be used in order to solve fluid dynamic problems by integration over time.

#### 2.2.1. Navier-Stokes equations in Lagrangian form

Navier-Stokes equations are widely used for studying fluid dynamics. These equations are based on fundamental laws of conservation of mass, and momentum. Two different approaches can be used in derivation of these equations, the Eulerian description and the Lagrangian description. In Eulerian specification, fluid motion is studied through a laboratory frame of reference. In Lagrangian approach, observer follows a fluid parcel as it moves within the domain. The nature of these two methods leads to usage of total derivatives in Lagrangian representation where
differentiations have both local and convective variation inside them. In SPH, using the equations in Lagrangian form is more convenient.

In the Lagrangian frame work, Navier-Stokes equations for conservation of mass and momentum of Newtonian fluids can be written as

**Continuity equation**

$$\frac{D \rho}{Dt} = -\rho \nabla \cdot V$$  \hspace{1cm} (16)

with the velocity vector in the form of $V = u \hat{i} + v \hat{j}$ on a two dimensional Cartesian domain.

**Momentum equation**

$$\frac{Du}{Dt} = \frac{1}{\rho} \left( \frac{\partial \sigma^{xx}}{\partial x} + \frac{\partial \sigma^{xy}}{\partial y} \right)$$

$$\frac{Dv}{Dt} = \frac{1}{\rho} \left( \frac{\partial \sigma^{yx}}{\partial x} + \frac{\partial \sigma^{yy}}{\partial y} \right)$$  \hspace{1cm} (17)

where

$$\sigma^{xx} = -p + \tau^{xx}$$

$$\sigma^{yy} = -p + \tau^{yy}$$

$$\sigma^{xy} = \tau^{xy}$$

with $p$ being the isotropic pressure and $\tau$ being the viscous stress in the form of (for Newtonian fluids)

$$\tau^{xx} = \mu \left\{ \frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} - \frac{2}{3} (\nabla \cdot V) \right\}$$

$$\tau^{yy} = \mu \left\{ \frac{\partial v}{\partial y} + \frac{\partial v}{\partial y} - \frac{2}{3} (\nabla \cdot V) \right\}$$

$$\tau^{xy} = \mu \left\{ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right\}$$

where $\mu$ is the dynamic viscosity.
2.2.2. SPH representation of continuity equation

The governing equations presented in the previous section can be solved and approximated through the particle approximation methods discussed earlier. For creating the approximation of continuity equation, the $\nabla \cdot \mathbf{V}$ part in equation (16) can be approximated using equation (15)

\[
(\nabla \cdot \mathbf{V})_{@x_0} \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{V}_{@x_j} \cdot \mathbf{V}^* W(x_j - x_0, h) \tag{18}
\]

Therefore substituting equation (18) into equation (16) leads to

\[
\left(\frac{D\rho}{Dt}\right)_{@x_0} \approx -\rho_{@x_0} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{V}_{@x_j} \cdot \mathbf{V}^* W(x_j - x_0, h) \tag{19}
\]

As was seen in particle approximation before, it can be assumed that a particle like particle $i$ is located on $x_0$ and all the related values to this specific particle calculated at this point are from its neighboring particles, $j$. Taking this into account, equation (19) can be rewritten as

\[
\frac{D\rho_i}{Dt} \approx -\rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{V}_j \cdot \mathbf{V}_i W(x_j - x_i, h) \tag{20}
\]

$\mathbf{V}^*$ is shown as $\mathbf{V}_i$ as it is calculated with respect to $x_0$ which is the location of particle $i$. By inserting a constant function like $f(x) \equiv 1$ into equation (15), the following identity can also be derived

\[
\nabla \cdot 1 = 0 \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{V}_i W(x_j - x_i, h) \tag{21}
\]

By multiplying the quantity $\rho_i \mathbf{V}_i$ to both sides of equation (21)

\[
0 \approx \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{V}_i \cdot \mathbf{V}_i W(x_j - x_i, h) \tag{22}
\]

$\mathbf{V}_i$ was taken inside the summation as it is not varying with respect to $j$. Adding equation (22) to equation (20) gives
\[
\frac{D\rho_i}{Dt} \approx \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (\mathbf{V}_i - \mathbf{V}_j) \cdot \mathbf{v}_i W(x_j - x_i, h)
\]

This approach is known as continuity density. In equation (23), the usage of velocity in the relative form of \( \mathbf{V}_i - \mathbf{V}_j \) reduces the particle inconsistency problem [18]. Also adding \( \mathbf{V}_i \) would ensure that the divergence vanishes for a uniform velocity distribution [19]. A disadvantage of this approach however would be the fact that it does not conserve mass exactly [20]. There are other methods also available for calculation of density variation, like summation density approach. In summation density, equation (6) is discretized using the particle approximated methods mentioned previously to turn into

\[
\int_{\phi} f(x) W(x - x_i, h) \, dx \approx f(x_i) \approx \sum_{j=1}^{N} f(x_j) W(x_j - x_i, h) \Delta V_j
\]

\[
= \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) W(x_j - x_i, h)
\]

(24)

By taking \( f(x) \) to be the density function, equation (24) gives

\[
(\rho)_0 \approx \rho_i \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} W(x_j - x_i, h) \Rightarrow
\]

\[
\rho_i \approx \sum_{j=1}^{N} m_j W(x_j - x_i, h)
\]

(25)

The density summation approach, equation (25), would conserve mass exactly (in case no particles are added or lost in the domain). This can be roughly demonstrated by assuming all particles having equal masses and volumes (\( \Delta V_i = \Delta V_j = \Delta V ) \). The mass of the entire domain can be computed using density from equation (25)

\[
\sum_{i=1}^{N} \rho_i \Delta V_i = \sum_{i=1}^{N} \left( \sum_{j=1}^{N} m_j W(x_j - x_i, h) \right) \Delta V_i
\]
In above equations, based on the particle approximation of equation (5), the following simplification is used

\[ \sum_{j=1}^{N} W(x_j - x_i, h) \Delta V_j \approx 1 \]  \hspace{1cm} (27)

In the work presented here, the summation density approach with some modifications is used. This is discussed in more details in section 3.2.2.

2.2.3. SPH representation of momentum equation

By performing the same procedure mentioned in 2.2.2, momentum equations in the lagrangian form of equation (17) can be rewritten using the particle approximation method. For particle \( i \) located at the end of \( x_0 = X_i \hat{i} + Y_i \hat{j} \) vector on a two dimensional domain (figure 2.3), momentum equation can be written as

\[
\begin{align*}
\frac{Du_i}{Dt} & \approx \frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \left[ (\sigma_{j}^{xx} + \sigma_{i}^{xx}) \frac{\partial W(x_j - x_i)}{\partial X_i} + (\sigma_{j}^{xy} + \sigma_{i}^{xy}) \frac{\partial W(x_j - x_i)}{\partial Y_i} \right] \\
\frac{Dv_i}{Dt} & \approx \frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \left[ (\sigma_{j}^{yy} + \sigma_{i}^{yy}) \frac{\partial W(x_j - x_i)}{\partial X_i} + (\sigma_{j}^{yx} + \sigma_{i}^{yx}) \frac{\partial W(x_j - x_i)}{\partial Y_i} \right]
\end{align*}
\]  \hspace{1cm} (28)

where

\[
\begin{align*}
\sigma_{j}^{xx} & = -p + \tau^{xx} \\
\sigma_{j}^{yy} & = -p + \tau^{yy} \\
\sigma_{j}^{xy} & = \tau^{xy} \\
\sigma_{j}^{yx} & = \tau^{yx}
\end{align*}
\]

with \( p \) being the isotropic pressure and the viscous stress, \( \tau \) in the form of
Detailed discussions of these equations can be found in [18, 20]. A disadvantage of this formulation is the nested loops which would increase computational costs excessively. One of the more computationally effective methods is presented by Monaghan [20]. This method which allows velocity variation calculations in a single pass is mostly used in strong shock wave simulations. An artificial viscous term is added which not only takes into account the conversion of kinetic energy into heat at the shock front, but also prevents unphysical particle penetrations [18]. The momentum equation in this form would be

\[
\tau_{xx} = 2\mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (u_j - u_i) \frac{\partial W(x_j - x_i)}{\partial X_i} - \frac{2}{3} \mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (v_i - v_j) \cdot \mathbf{v}_i W(x_j - x_i, h)
\]

\[
\tau_{xy} = 2\mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (v_j - v_i) \frac{\partial W(x_j - x_i)}{\partial Y_i} - \frac{2}{3} \mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (v_i - v_j) \cdot \mathbf{v}_i W(x_j - x_i, h)
\]

\[
\tau_{yx} = \mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (v_j - v_i) \frac{\partial W(x_j - x_i)}{\partial X_i} + \mu_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} (u_j - u_i) \frac{\partial W(x_j - x_i)}{\partial Y_i}
\]

where \( \Pi_{ij} \), the artificial viscosity term is in the form of

\[
\Pi_{ij} = \begin{cases} 
-\alpha \frac{\mu_{ij}}{\bar{\rho}_{ij}} + \beta \frac{\bar{\mu}_{ij}^2}{\bar{\rho}_{ij}} & (V_i - V_j) \cdot (x_i - x_j) < 0 \\
0 & \text{otherwise}
\end{cases}
\]

with

\[
\bar{\mu}_{ij} = \frac{h(V_i - V_j) \cdot (x_i - x_j)}{(x_i - x_j)^2 + 0.01 h^2}
\]

\[
\bar{\rho}_{ij} = \frac{1}{2} (\rho_i + \rho_j)
\]
In above equations, $\alpha$, $\beta$, and $c$ are constants. This formulation conserves angular momentum which is useful in high velocity and shock wave simulations. Morris et al. [21] have argued that utilizing equation (29) in low Reynolds simulations would produce unrealistic results. Instead, they have proposed the following formulation

$$\frac{du_i}{dt} = -m_i \sum_{j=1}^{N} \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{\partial w(x_j-x_i)}{\partial x_i} +$$

$$m_i \sum_{j=1}^{N} \left( \frac{\mu_i + \mu_j}{\rho \rho_j} \right) (u_i - u_j) \left[ \frac{x_i-x_j}{\sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}} \frac{\partial w(x_j-x_i)}{\partial x_i} + \frac{y_i-y_j}{\sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}} \frac{\partial w(x_j-x_i)}{\partial y_i} \right]$$

$$\frac{dv_i}{dt} = -m_i \sum_{j=1}^{N} \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{\partial w(x_j-x_i)}{\partial y_i} +$$

$$m_i \sum_{j=1}^{N} \left( \frac{\mu_i + \mu_j}{\rho \rho_j} \right) (v_i - v_j) \left[ \frac{x_i-x_j}{\sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}} \frac{\partial w(x_j-x_i)}{\partial x_i} + \frac{y_i-y_j}{\sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}} \frac{\partial w(x_j-x_i)}{\partial y_i} \right]$$

(32)

In these equations, linear momentum is conserved exactly while angular momentum is only approximately conserved [21].

### 2.3. Equation of state

The conservation equations mentioned above contain unknown variables of $p$, $u$, $v$ and $\rho$. For closing these set of equations, a supplementary equation is needed. For the case of ideal gas, ideal gas law is usually chosen as it connects $p$ to $\rho$ (and temperature). This equation can be in the form of

$$p = (\gamma - 1)\rho e$$

(33)

With $\gamma = C_p/C_v$ being the adiabatic index, $e = C_v T$, the internal energy per unit mass,. $C_p$ and $C_v$ the specific heats at constant pressure and volume, respectively.

For liquids like water, due to their large value of sound speed, if the actual equation of state is used, a very small time step has to be chosen for stability reasons (by the CFL condition) [21].
Moreover, movement of particles in SPH is because of small pressure variations locally. As these pressure variations are calculated from density variations in equation of state, a quasi-incompressible fluid must be considered instead in which the fluid is allowed to have limited compressible characteristics. This equation is chosen in the following form as suggested by Bachelor [22]

\[ p = p_0 \left( \frac{p}{p_0} \right)^\gamma - 1 \]  \hspace{1cm} (34)

where \( p_0 \) and \( p_0 \) are the initial pressure and density. Calculating an initial speed of sound \( c_0 \) from above equation would lead to

\[ c^2 = \frac{\partial p}{\partial \rho} = \frac{p_0 \gamma}{p_0} \left( \frac{\rho}{p_0} \right)^{\gamma-1} \]

\[ c_0^2 |_{\rho=p_0} = \frac{p_0 \gamma}{p_0} \]  \hspace{1cm} (35)

Therefore \( p_0 \) in equation (34) can be replaced from (35) resulting in

\[ p = \frac{c_0^2 \rho_0}{\gamma} \left( \frac{\rho}{p_0} \right)^\gamma - 1 \]  \hspace{1cm} (36)

Monaghan [12] has proposed that from the momentum equation, variation of density can be shown to be in the form of \( \delta \rho / \rho = vL/c^2 \tau \) with \( L \) and \( \tau \) being the typical length and time scales and \( v \) is the typical fluid velocity. By assuming \( L/\tau \propto v \), variations of density would then be comparable with Mach number, \( \delta \rho / \rho \propto v^2 / c^2 = M^2 \). Therefore, for keeping variations of density below a desirable number like 1%, Mach number should be less than 0.1, i.e. the imposed speed of sound should be 10 times larger than the maximum expected velocity in the fluid domain (\( c_0 \propto 10v_0 \)). For instance, Monaghan [12] has suggested for the case of dam break with a height of \( H \), since the maximum velocity can be predicted to be on the order of \( v^2 = 2gH \), the speed of sound can be taken as \( c_0 = \sqrt{200gH} \). The adiabatic index (\( \gamma \)) for the case of incompressible fluids is also usually chosen between 1 to 7 in SPH literature based on numerical preferences.
2.4. Smoothing functions

Different kinds of smoothing functions are employed in SPH for different applications. These functions can be chosen arbitrarily as long as they satisfy all the characteristics discussed in section 2.1.1. In other words, the smoothing kernel should be normalized

\[ \int_{-\infty}^{\infty} W(x - x_0, h) \, dx = 1 \]  

(37)

and have a compact support domain,

\[ W(x - x_0, h) = 0 \quad \text{if} \quad |x - x_0| \geq kh \]  

(38)

It should also be an even function of \((x - x_0)\), as discussed for equation (8) which satisfies

\[ \lim_{h \to 0} W(x - x_0, h) = \delta(x - x_0) \]  

(39)

Although many functions can be found with these properties, only some have the practical value for use in SPH simulations. For instance, a tent function satisfies all the above criteria while it is unstable under positive stresses and demands a non-physical speed of sound for negative stresses [23].

Gaussian kernels are among the first kernels suggested for use in SPH

\[ W(r, h) = \alpha^* \exp\left(-\frac{r^2}{h^2}\right) \]  

(40)

here \(\alpha^*\) is a constant which can be derived based on the conditions mentioned above for each of one, two, or three dimensional domains and is \(1/\pi^{\frac{3}{2}}h\), \(1/\pi h^2\), and \(1/\pi^{\frac{3}{2}}h^3\) respectively. \(r\) is also defined as the distance between the two particles of \(i\) and \(j\) in the form of \(r = |x_i - x_j|\). The advantage of using a Gaussian kernel is that its derivative and Fourier transform are still a Gaussian. This plays an important role in stability properties of this kernel (discussed in detail in [23, 24]). The main disadvantage of this kernel is not satisfying the compact condition of equation (38), or in other words having all particles contributing to calculated values of a single particle, although many values might be near zero and negligible. This increases the time consumption of the solver. To overcome this, it has been suggested that this kernel still gives
accurate values when it is imposed on particles with a neighborhood radius of at least $3h$ or larger, which means $W(r, h)$ can be neglected when $\frac{r}{h} \geq 3$ [25]. Hence, the kernel is constructed on $(-\infty, +\infty)$ but only used on a $\frac{r}{h} \leq 3$ domain [26].

There are also other types of kernels available that are based on polynomials. Johnson et al. [27] suggested a quadratic kernel in the form of

$$W(r, h) = \alpha^* \left\{ \begin{array}{ll}
\frac{3}{16} \left(\frac{r}{h}\right)^2 - \frac{3}{4} \left(\frac{r}{h}\right) + \frac{3}{4} & \text{if } 0 \leq \frac{r}{h} \leq 2 \\
0 & \text{if } \frac{r}{h} \geq 2
\end{array} \right. \quad (41)$$

with $\alpha^*$ being $1/h$, $2/\pi h^2$ and $5/4\pi h^3$ in one, two and three dimensional domains respectively. The derivative of this kernel always increases when particles get closer and always decreases when particles are moving away. This property allows this kernel to perform more realistically when compared with cubic kernel derivatives and has been reported to eliminate compressive instabilities [27]. Another popular kernel is the cubic spline constructed based on cubic polynomials suggested by Monaghan et al. [25] in the form of

$$W(r, h) = \alpha^* \left\{ \begin{array}{ll}
\frac{2}{3} - \left(\frac{r}{h}\right)^2 + \frac{1}{2} \left(\frac{r}{h}\right)^3 & \text{if } 0 \leq \frac{r}{h} < 1 \\
\frac{1}{6} \left(2 - \frac{r}{h}\right)^3 & \text{if } 1 \leq \frac{r}{h} < 2 \\
0 & \text{if } \frac{r}{h} \geq 2
\end{array} \right. \quad (42)$$

with $\alpha^*$ being $1/h$, $15/7\pi h^2$ and $3/2\pi h^3$ in one, two and three dimensional domains respectively. Satisfying the compact condition of equation (38) makes this kernel computationally much more effective as less number of particles would be engaged in the process of calculating certain properties of each particle. However, it has been shown the usage of this kernel would produce errors in dispersion relation for linear waves [23, 24].

Similarly, kernels of quartic (fourth order polynomial) and quintic (fifth order polynomial) can be constructed. For instance the quintic kernel would be
With $\alpha^*$ being $120/h$, $7/478\pi h^2$ and $3/359\pi h^3$ in one, two and three dimensional domains respectively. These higher order kernels are more stable as they better estimate the Gaussian kernel [18]. The main disadvantage of these high order kernels would be having more particles contributing to calculations which would make these kernels computationally more expensive.

### 2.5. Boundary treatments

There are three main issues regarding a solid wall boundary in SPH that need to be addressed. Initially, since particles near boundary would have fewer neighboring particles than those located in the bulk of the fluid, if density is being calculated from equation (25), the densities calculated for these particles would be lower than rest of particles. Moreover particles with comparable higher velocities might penetrate through the wall which might have no physical explanation. Applying boundary conditions like no slip or Neumann on the walls also require special considerations.

For preventing particles from penetrating into boundaries, Monaghan [12] suggested locating particles on the wall boundary. These particles are stationary and do not contribute to density or momentum calculations of particles in the bulk of fluid. Instead, they insert a repulsive force to the particles getting close to the boundary. This force is usually chosen in the form of Lennard-Jones potential as

$$ f(r) = \begin{cases} 
D \left( \left( \frac{r_0}{r^*} \right)^{12} - \left( \frac{r_0}{r^*} \right)^6 \right) \frac{r^*}{r^{*2}} & r^* < r_0 \\
0 & r^* \geq r_0 
\end{cases} $$

(44)
Where \( \mathbf{r}^* \) is the position vector from the boundary particle to the particle receiving the repulsive force and \( r_0 \) is the cutoff distance. Particles closer than \( r_0 \) to each boundary particle would receive the repulsive force from that specific boundary particle. \( D \) is chosen based on the physical properties and expected velocity field in the fluid domain.

Morris et al. [21] have suggested another procedure which is also useful for enforcing boundary conditions like no slip. In this method, ghost particles are located inside the wall. These particles contribute to density and pressure calculations of real fluid particles. In this way, all real fluid particles would have enough neighbors in their supporting domain. Morris et al. have reported that better results would be captured if the density and pressure of these ghost particles are also evolved with time instead of being kept constant. The velocity assigned to these particles contributes to the momentum calculations of real fluid particles by appearing in equation (32). In this procedure, for enforcing a no-slip boundary condition, the velocity of the ghost particle is assigned based on the real particle that the ghost particle is going to contribute to it. For each real particle, a normal distance to the boundary, \( d_{\text{real}} \), is found. This normal is used to construct a tangent plane to the boundary as illustrated in figure 2.4. The velocity of each ghost particle in the neighborhood of this real particle is then calculated using the distance of the ghost particle to this tangent plane, \( d_{\text{ghost}} \), in the form of

\[
\mathbf{v}_{\text{ghost}} = -\frac{d_{\text{ghost}}}{d_{\text{real}}} \mathbf{v}_{\text{real}}
\]  

(45)

Since a real particle can get close to the wall, or in other words \( d_{\text{real}} \) could approach to values near zero, \( \mathbf{v}_{\text{ghost}} \) might possess large values. In order to avoid that, Morris et al. have suggested limiting \( \frac{d_{\text{ghost}}}{d_{\text{real}}} \) to a value of 0.5 obtained from numerical simulations. A similar boundary treatment procedure similar to the method of Morris et al. has also been introduced by Zhu et al. [28].
The main disadvantage of the method of Morris et al. is the necessity for finding the tangent planes to the boundary. This either demands knowledge of the geometry of the wall boundaries prior to developing the solver, or in a more general case, having a process which takes care of defining an estimate for geometry of the boundaries. Moreover, obtaining the tangent to the wall and then calculating distances from that tangent can also be computationally inefficient. Another more recent method in treating boundaries in SPH which avoids these computations is proposed by Holmes et al. [29]. In this method, a state-specific particle density is defined for particle $i$ in the following way

$$
\bar{n}_i = \sum_{j=1}^{N} \delta^* W(x_j - x_i, h)
$$

with $\delta^*$ being Kronecker delta defined to be ‘1’ when particle $j$ in the neighborhood of particle $i$ is of the same state of particle $i$, i.e. they are both located inside the wall or are both located inside the fluid; otherwise $\delta^*$ is ‘0’.

In addition, a second particle density is also defined, which includes all the neighbors of particle $i$ in the form of

$$
n_i = \sum_{j=1}^{N} W(x_j - x_i, h)
$$

$Figure 2.4. Calculation of ghost particle velocities based on real particle velocities for constructing a no-slip boundary condition$
Hence a ratio for particle $i$ can be defined in the form of

$$\chi_i = \frac{n_i}{\bar{n}_i} \quad (48)$$

It is apparent that for the case of a straight wall boundary, if a fluid or wall particle is located right on the boundary, $\chi_i$ would be equal to 0.5 as half of the neighbors of this specific particle would be located in its same state. According to compact condition of equation (7), neighborhood of a particle would have a radius of $kh$, as can be seen in figure 2.3. While the distance of a particle to the wall varies between 0 and $kh$, the value of $\chi_i$ would vary between 0.5 on the wall (figure 2.5.c) to 1 when the particle is located at the distance of $kh$ from the wall (figure 2.5.a). For the rest of distances larger than $kh$, $\chi_i$ would have a value of 1, as all particles in the neighborhood of particle $i$ would be of same state of this particle.

Like in the method of Morris et al., the velocity of the ghost particles can be calculated from the real particle that they are contributing to it. By using equation (48), this velocity as suggested by Holmes et al. can be defined in the form of

$$\mathbf{v}_{ghost} = \frac{kh}{0.5} \frac{(\chi_{ghost} - 0.5)}{\max\left(\frac{kh}{0.5} (\chi_{real} - 0.5), \frac{\sqrt{3}}{4} h\right)} \mathbf{v}_{real} \quad (49)$$

Figure 2.5. a) when fluid particle only has fluid neighbors ($\chi_i = 1$), b) when fluid particle has both fluid and wall neighbors ($0.5 \leq \chi_i \leq 1$), c) when fluid particle has equal fluid and wall neighbors ($\chi_i = 0.5$)
More discussions on boundary treatments can be found in the works of Yildiz et al. [30] and Zhu et al. [28].

2.6. Time integration

When the variations in density and velocity of each particle have been calculated, a time marching method should be utilized in order to impose the obtained values on velocity and position of particles. Particles in SHP are usually moved according to

$$\frac{Dx_i}{Dt} = v_i$$  \hspace{1cm} (50)

In XSPH proposed by Monaghan [31] particles are moved instead with the use of

$$\frac{Dx_i}{Dt} = v_i + \epsilon \sum_{j=1}^{N} 2m_j \frac{V_j - V_i}{\rho_j + \rho_j} W(x_j - x_i, h)$$  \hspace{1cm} (51)

where $\epsilon$ is a predefined constant between zero and one. The idea behind XSPH is to move particle $i$ with a modified velocity which is closer to the average velocity in the neighborhood of particle $i$ to avoid unwanted particle penetrations.

When all governing equations in SPH are chosen, any technique such as velocity-verlet [32, 33], Leapfrog [20, 34], predictor-corrector [20], Runge-Kutta and etc. can be used for performing numerical integration. The time step in each of these schemes should be selected correctly. An acceptable choice of time step conserves the total energy within 0.5% over 400 steps [20].

Explicit time marching schemes are stable under the Courant–Friedrichs–Lewy condition (CFL). The logic behind this limiting condition is that the speed of numerical propagation (dependent on the choice of time step, $\Delta t$) should exceed the speed of physical propagation. The CFL condition in SPH is shown as

$$\Delta t \leq \frac{h}{c}$$  \hspace{1cm} (52)
with $c$ being the speed of sound in domain. $\frac{h}{c}$ in equation (52) is the ratio of smallest particle resolution (usually equal to $h$) to speed of sound, $c$. Some publications like Morris et al. [21] have taken only 25 percent of the $\Delta t$ in equation (52) as a safety factor which makes sure the particle moves only a fraction (0.25) of the smallest particle resolution (usually equal to $h$) per time step [35].

Morris et al. [21] also suggested another criterion based on the viscous diffusion in the form of

$$\Delta t \leq \frac{h^2}{\nu}$$

(53)

where $\nu$ is the kinetic viscosity. Equation (53) can also be multiplied by a safety factor of 0.125 as suggested by Morris et al.. Another condition mentioned in literature is in the form of [20, 21, 31]

$$\Delta t \leq \left(\frac{h}{f}\right)^{\frac{1}{2}}$$

(54)

with $f$ being the magnitude of force per unit mass inserted on each particle. Again a safety factor of 0.125 can be imposed on the condition in equation (54). These criteria might get modified when different forms of governing equations in SPH are used.
Chapter 3
Drop Formation

For simulations containing formation and evolution of drops, having a reliable surface tension model is necessary [36]. There are different methods available for implementing surface tension in SPH. A liquid drop can be modeled in SPH either in the single-phase SPH schemes which are normally used for studying free surface fluid flows or in the multi-phase SPH schemes which allow the simulation of separate phases at the same time. In this chapter, each of these methods is described and challenges towards drop simulations are investigated.

3.1. Drops in Single-phase SPH

Free surface flows studies have been of long interest in SPH simulations. Dam breaks [12], water waves [37, 7], and movements of floating objects on the surface of water like ships [38] are among many cases studied in this area. Monaghan [12] has studied free surface flow simulations in SPH in detail. For the cases of large fluid domains, like dam breaks, simulations usually do not need to take into account the surface tension forces acting on the fluid interfaces. For simulations containing formation and evolution of drops, having a reliable surface tension model is necessary [36].

One of the methods which can be added to Single-phase SPH is proposed by Nugent et al. [39]. This method is based on van der Waals equation of state. In this algorithm, density and momentum conservations can still be calculated from previously discussed methods in Chapter 2 (equations similar to equations (25) and (28)). Instead of closing these equations with the equations of state introduced before, a van der Waals equation of state is used. Starting from van der Waals equation in the form of

\[ p = \frac{Nk_B T}{V^* - Nb} - \left( \frac{N}{V^*} \right)^2 a \]  

with \( k_B \) being the Boltzmann’s constant, \( T \) the temperature, \( N \) the number of particles and \( V^* \) the volume of the container. \( a \) and \( b \) are measures of the attractions between particles and the
volume each particle occupies, respectively. By substituting \( mN/V^* = \rho \), where \( m \) is the mass of each particle, into equation (55),

\[
p = \frac{\rho kT}{1 - \rho b} - \bar{a} \rho^2
\]  

(56)

where \( k = k_B/m \), \( \bar{a} = a/m^2 \) and \( b = b/m \). The following equation of state is used to calculate pressure of each particle from its density and temperature. By inserting equation (56) into conservation of momentum equations, the term \( -\bar{a} \rho^2 \) would create acceleration for each particle in the form of

\[
\frac{DV_i}{Dt} = 2\bar{a} \sum_{j=1}^{N} m_j \nabla_i W(x_j - x_i, h)^{h \rightarrow H}
\]  

(57)

These attractive forces tend to be canceled by each other in the bulk of fluid, while near the interface, in a strip of width \( H \), these forces remain pointing towards the denser phase analogous to a surface tension force acting on the interface. The denser fluid has a denser positioning of particles or a larger particle mass compared to the phase outside the drop. The van der Waals method can be used for simulating drops as a single phase, even for drops when they are in equilibrium with their own vapor phase. As is suggested by Nugent et al. [39], the radius of the smoothing function, \( W \), used in equation (57) should be taken larger (\( h \rightarrow H \)) than the radius of smoothing functions in other conservation equations. The value of \( H = 2h \) is usually chosen in literature. The temperature in equation (56) can also be evolved throughout the simulation with the help of the conservation of internal energy (see [39] for more detail).

One of the disadvantages of this method is the large number of particles that would contribute to equation (57). Particle \( i \) in equation (57), due to the increase in the radius of smoothing function (\( h \rightarrow H \)), would have more neighbors. Another disadvantage is the fact that this equation is calculated all over the domain. In other words, the force calculated by this formulation is inserted on all particles inside the domain rather than only the particles forming the surface of the drop (for the particles in the bulk of the drop, these forces would cancel each other out and the effect of surface tension would be left on the particles near the surface). The resulting increase in computational costs can be considerable. Additionally, since surface tension coefficient cannot be inserted directly into these equations and is calculated as a byproduct of the model from
Laplace equation, the method needs to be calibrated if specific surface tension coefficients are needed to be modeled. Moreover, resulting surface tension coefficient would be mesh-dependent [40]. It is also crucial to mention that liquid drops formed using this method would suffer from tensile instability. In the presence of this instability, which is mostly due to existence of attractive forces introduced from pressure equation, particles tend to form small clusters during reaching equilibrium. Tensile instabilities have been studied in detail and beneficial treatments have been suggested by studies like Meleán et al. [41] and Gray et al. [42]. However, these treatments would increase the computational costs of the method.

Tartakovsky et al. [32] used a similar method with some changes in order to reduce the number of particles engaged in surface tension force calculations. In this model, the van der Waals equation of state in the form of equation (56) is still used. Instead of separating the attractive forces in the form of equation (57) which needed to be treated with larger radius for smoothing function (\(H\)), a pairwise interaction force is added to the momentum equation in the form of

\[
\frac{DV_i}{Dt} = \frac{1}{m_i} \sum_{j=1}^{N} F_{ij}
\]  

(58)

Where the force between each two particles is defined in the following way

\[
F_{ij} = \begin{cases} 
S_{ij} \cos \left( \frac{1.5\pi}{3h} |x_j - x_i| \right) \frac{x_j - x_i}{|x_j - x_i|} & |x_j - x_i| \leq h \\
0 & |x_j - x_i| > h
\end{cases}
\]

(59)

The force in equation (59) would be in the form of repulsion for distances below \(h/3\) and attraction for distances between \(h/3\) to \(h\). \(S_{ij}\) in equation (59) can be used as an adjusting coefficient for the amount of forces acting between particle pairs. Tartakovsky et al. have shown that by choosing different interaction strengths (different \(S_{ij}\)) for particles near the wall, wetting behaviors on the solid boundaries can be modeled. Although results proposed by Tartakovsky et al. are promising, their method still needs calibration and can be mesh-dependent [40].
3.2. Drops in Multi-phase SPH

Many attempts have been made in simulating multi-phase fluid phenomena using SPH. Some of these multi-phase studies like work of Colagrossi et al. [43] have been performed without any specific treatments for modeling physical surface tension effects. Although these simulations produce acceptable results for interfacial flows, for cases of droplet formation especially in small scale cases, having a reliable surface tension treatment algorithm is reported to be necessary [43].

Hence, other multiphase studies have employed different methods for adding surface tension effects like the proposed methods of Morris [44] and Hu et al. [1, 11, 45]. Unlike the methods described in the previous section for single-phase SPH, these methods are usually based on macroscopic surface tension models. In these models, by estimating the curvature of the surface, surface tension can be applied as a continuous force near the interface [40].

3.2.1. Interface tracking

Any multiphase scheme that is chosen for handling surface tension effects should also be able to keep track of the interface between different phases. There are a variety of choices that can be used as an interface tracking method. These methods can be generally categorized into three groups: surface tracking methods, volume tracking methods, and moving mesh methods [46].

Surface tracking methods are considered to be simple and straightforward from implementation aspects. In these methods, only markers located on the surface are usually tracked. The interface between these markers is needed to be approximated by interpolation, like by using piecewise polynomial functions. Surface tracking methods can sometimes be misleading, especially when the interface geometry is constantly changing through computation. For instance, the interface can be tracked using a height function, tracking the distance of each marker to a reference line. If the interface experiences some drastic changes, these height functions can become multi valued for some points. The fact that interfaces can interact with each other (merge together or get separated) would even raise the complexity of the surface tracking methods. It has been suggested that, under interaction conditions, volume tracking methods can be used instead, especially for 3D cases [46].
In Volume tracking methods, phases are treated as separate solutions. These solutions can be tracked even by the fraction of each phase inside each domain cell (like those used in Volume of Fluid method), or by having particles assigned to each phase carrying fluid characteristics. The latter approach can be used in Lagrangian methods like SPH. In other words, each SPH fluid particle belongs to a specific fluid phase and remains part of that phase throughout the computation.

The volume tracking methods can be computationally more expensive than other methods, as particles are needed to sweep the whole domain while they could have only been located near the actual interface. On the other hand, main advantage of this method is the fact that having several phases at the same time in the domain would only demand adding separate particle types in charge of tracking each phase.

Moving mesh methods are not discussed here due to the mesh free nature of SPH. In these methods, mesh cells are locally adjusted to be aligned with the interface (see [46] for more details).

### 3.2.2. Continuum surface force (CSF)

After the interfaces between different phases have been located, a separate method should be used for taking surface tension effects on these interfaces into account. Continuum surface force (CSF) method proposed by Brackbill et al. [15] is one of the models that can be used for numerical simulation of surface tension force.

In the CSF model, each fluid phase is assigned a constant color function, $C$, which has a unit jump at each interface. The surface tension in the form of a force inserted on the interface (as a boundary condition) is then substituted by a volumetric force inserted across the interface obtained from [15]

$$
\lim_{h' \to 0} \iiint F_{sv} \, dV = \iint F_{sa} \, dA
$$

(60)
where \( F_{sa} \) is the surface tension force inserted on the interface which can be substituted by a volumetric force \( F_{sv} \) that is inserted over a transition region with the width of \( h' \) which contains the actual interface in the middle.

By using integral interpolations similar to the ones discussed in section 2.1.1, the \( F_{sv} \) in equation (60) can be approximated in the form of

\[
F_{sv} = \alpha \kappa \hat{n} \delta_s
\]  

(61)

where \( \alpha \) is the surface tension coefficient, \( \kappa \) the curvature of the phase interface, \( \hat{n} \) the unit normal which is perpendicular to the interface, and \( \delta_s \), surface delta function which makes sure \( F_{sv} \) vanishes outside the interface transition region (outside the width of \( h' \)). Surface tension is assumed to be constant and hence equation (61) does not consider Marangoni effects. The unit normal vector can be calculated from the gradient of the color function in the form of

\[
\hat{n} = \frac{\nabla c}{|\nabla c|}
\]  

(62)

The curvature of the interface, \( \kappa \), is then calculated form unit normal vector as

\[
\kappa = -\nabla \cdot \hat{n}
\]  

(63)

3.2.3. SPH implementation

From the models reviewed earlier, the CSF method is chosen for implementation in this study. The interface tracking method and CSF model can be modeled in SPH with different procedures which may vary in details. Morris [44], Hu et al. [1], Adami et al. [40], and Das et al. [17, 47] are among those who proposed various methods for using CSF in SPH. In the following section, due to preferences in the upcoming chapters, only the method of Hu et al. is reviewed.

SPH equations in the manner discussed in the previous chapters need to be modified to be able to capture all the phenomena related to multiphase fluid. Isothermal and incompressible Navier-Stokes equations in a Lagrangian framework are
\[
\frac{dp}{dt} = -\rho \nabla \cdot \mathbf{V} \tag{64}
\]

\[
\frac{d\mathbf{V}}{dt} = \frac{1}{\rho} \left[ -\nabla p + \mu \nabla^2 \mathbf{V} + \mathbf{F}_{sv} + \mathbf{F}_b \right] \tag{65}
\]

where \( \frac{d}{dt} \) is the total derivative (\( \frac{d\varphi}{dt} = \frac{\partial\varphi}{\partial t} + \mathbf{V} \cdot \nabla \varphi \)), \( \mathbf{F}_b \) represents external body forces such as gravity. The surface tension force, \( \mathbf{F}_{sv} \), as discussed previously is approximated based on the Continuum Surface Force (CSF) model of Brackbill et al. [15], and for the case of constant surface tension is given by equation (61).

The density of particle \( i \), is calculated based on the summation density method rather than continuity approach (using equation (25) instead of equations (23) and (64)). Equation (25) is also modified to get the form of

\[
\rho_i = m_i \sum_j W_{ij} = m_i \sigma_i \tag{66}
\]

The advantage of using equation (66) is that density of particle \( i \) only depends on the mass of this particle (\( m_i \) is used instead of mass of all neighboring particles, \( m_j \)). Therefore density is not smoothed near the regions were two phases in a multiphase environment meet as the mass of particles in different phases would differ from each other. This gives equation (66) the ability of reproducing sharp density variations between the two phases. In this way, each particle treats all its neighbors as if they have the same rest density and mass as itself [48]. As Hu et al. [1] suggested, the suitable particle-averaged spatial derivative for multiphase simulations for a smoothed variable \( \psi \) can be derived in the form of

\[
\nabla\psi_i = \sum_j \left( \frac{\psi_i}{\sigma_i^2} + \frac{\psi_j}{\sigma_j^2} \right) \sigma_i \frac{\partial W_{ij}}{\partial r_{ij}} \mathbf{e}_{ij} \tag{67}
\]

where \( \mathbf{e}_{ij} \) is the normalized vector from particle \( i \) to \( j \). Using equation (67) for calculation of the pressure term in equation (65), the acceleration caused by pressure for particle \( i \) is obtained by

\[
\mathbf{F}_i^p = -\frac{1}{m_i} \sum_j \left( \frac{p_i}{\sigma_i^2} + \frac{p_j}{\sigma_j^2} \right) \frac{\partial W_{ij}}{\partial r_{ij}} \mathbf{e}_{ij} \tag{68}
\]
Considering two phases \( k \) and \( l \), the viscous term in equation (65) can be calculated from \( F_v = \nabla \cdot \Pi \) where \( \Pi = \mu(\nabla \mathbf{V} + \nabla \mathbf{V}^T) \). The averaged shear stress between particles can be approximated \([49]\)

\[
\bar{\Pi}^y_{ij} = \frac{2\mu^k\mu^l}{r_{ij}(\mu^l + \mu^k)} \left( e_{ij} \mathbf{V}_{ij} + \mathbf{V}_{ij} e_{ij} \right) \tag{69}
\]

Combining equations (69) and (67), the viscous force in equation (65) can be rearranged in the form of \([1]\)

\[
\mathbf{F}_v = \frac{1}{m_i} \sum_j 2\mu^k\mu^l \left( \frac{1}{\sigma_i^2} + \frac{1}{\sigma_j^2} \right) \frac{\mathbf{V}_{ij} \partial \mathbf{W}_{ij}}{r_{ij} \partial r_{ij}} \tag{70}
\]

To calculate the surface force using CSF model, a color function \( C_i^s \) can be defined with a value of unity when particle \( i \) belongs to the arbitrary phase \( s \) and otherwise zero. Using equation (67), the gradient of this color function for particle \( i \) of phase \( k \) can be written as

\[
\nabla C_i^{k,l} = \sigma_i \sum_j \left( \frac{C_i^k}{\sigma_i^2} + \frac{C_j^l}{\sigma_j^2} \right) \frac{\partial \mathbf{W}_{ij}}{\partial r_{ij}} e_{ij} \tag{71}
\]

Considering equation (61) in the tensor notation of

\[
\mathbf{F}^{sv} = \nabla \cdot \mathbf{\Pi}^{sv} \tag{72}
\]

The surface stress tensor \( \mathbf{\Pi}^{sv} \) between the two phases of \( k \) and \( l \), can be written as \([50]\)

\[
\mathbf{\Pi}^{sv} = \alpha \left( \frac{1}{d} \mathbf{I} - \mathbf{n} \mathbf{n} \right) |\nabla \mathbf{C}| \tag{73}
\]

where \( d \) is the dimensionality parameter and \( \mathbf{I} \) the unit tensor. By defining \( \mathbf{n} = \nabla \mathbf{C} / |\nabla \mathbf{C}| \), equation (73) can be rewritten as

\[
\mathbf{\Pi}^{k,l}_i = \alpha^{k,l} \frac{1}{|\nabla \mathbf{C}_i^{k,l}|} \left( \frac{1}{d} \mathbf{I} |\nabla \mathbf{C}_i^{k,l}|^2 - \nabla \mathbf{C}_i^{k,l} \nabla \mathbf{C}_i^{k,l} \right) \tag{74}
\]

The total surface stress tensor can then be gained from the summation of tensors between particle \( i \) and other different phases in the form of
\[ \Pi_{iek}^{total} = \sum_{\forall l \neq k} \Pi_{i}^{k,l} \]  

(75)

By using equations (67) and (75), equation (72) for particle \( i \) becomes

\[ F^{sv} = \frac{1}{m_i} \sum_j \frac{\partial W_{ij}}{\partial r_{ij}} e_{ij} \left( \frac{\Pi_{i}^{total}}{\sigma_i^2} + \frac{\Pi_{j}^{total}}{\sigma_j^2} \right) \]  

(76)
As discussed in previous chapter, different approaches are available in SPH for implementing surface tension forces. Along with surface tension effects between two phases usually raises the question of proper treatment for having more than two separate phases, like having a wall boundary near the interface of a liquid and gaseous phase.

There are a variety of phenomena involved at this triple point. For the case of a moving contact line, the triple point forms a singularity. One of the proper treatments for this singularity at the contact line is applying a proper slip model. Using slip would allow the singular stresses near the contact line to be relaxed [3]. Slip models are not studied here for the moment.

Another subject of interest in the contact line is the contact angle that the three phases involved make with each other. This angle is mostly shown to be dependent on the three phase’s surface tension coefficients, and the velocity that the contact line is moving. In this chapter methods for implementing contact angle in SPH are studied and a proposed method is validated.

4.1. Contact angle in SPH

The measured contact angle from different methods mostly depends on the surface tension model that is being employed. Das et al. [17, 16] used CSF model for surface tension in their studies. They concluded that the resulting contact angle obtained by only applying the surface tension forces was not accurate enough and therefore suggested correcting the contact angle of the drop by repositioning the particles that form the contact line. After each time step, particles are repositioned to match the desirable angle and then, continuity and momentum equations are again satisfied to make sure the possible unphysical effects of the particles repositioning are minimized.

Another approach was introduced in the model of Hu et al. [1], where different surface tension coefficients are defined at the liquid-solid, gas-liquid, and gas-solid interfaces. At the triple point, these coefficients relate to one another by the Young-Laplace theory [51]
\[ \alpha^{lg} \cos \theta = \alpha^{sg} - \alpha^{sl} \tag{77} \]

In the case of a stationary droplet, which involves the three phases of liquid, vapor, and solid, using the three surface tension coefficients (\(\alpha^{lg}\), \(\alpha^{sg}\), and \(\alpha^{sl}\)) along with equation (75) guarantees an equilibrium contact angle close to what is expected from Young-Laplace theory, as previously reported by Hu et al. [1].

Results obtained using the method of Hu et al. are satisfying as the model is able to successfully reconstruct stationary contact angles between three phases based on the three defined surface coefficients. However, there are some disadvantages to this model. Specially in the reconstruction of a stationary or moving contact line on a wetted/non-wetted wall, only the surface tension between the liquid and gas phase seem to be of importance whereas in this model, also forces between the gas solid and liquid solid would be calculated that except for near the triple point, these forces tend to be canceled out. Calculation of these forces demands entering a whole new phase of solid boundary into relations calculating the surface tension which can increase run time. Another problem associated with this procedure is the lack for proper implementation of the dynamic contact angles as only the three surface coefficients play role in contact angle formation and by nature they are constant values related to the consisting material of each phase.

In this study, a different approach is introduced. This approach would be close to a combination of the works of Šikalo et al. [2] and Afkhami et al. [3] which have studied effects of dynamic contact angle on a Volume of Fluid (VOF) model. Here an effort is made to utilize similar approaches in available SPH models. In this method, the foundation of multiphase SPH fluid solver is constructed based on the multiphase model of Hu et al. [1] which has been previously reviewed in section 3.2.3. For the surface tension force calculation, in the method of Hu et al. the surface tension is calculated based on the gradient of the surface tension tensor in the form of

\[ \Pi^{sv} = \alpha \left( \frac{1}{d} I - \hat{n}\hat{n} \right) |\nabla \mathcal{C}| \tag{78} \]

which in the particle form can be written as
\[
\Pi^{k,l}_i = \alpha^{k,l} \frac{1}{|\nabla C^{k,l}_i|} \left( \frac{1}{d} \nabla C^{k,l}_i \cdot \nabla C^{k,l}_i \right) \tag{79}
\]

where the surface tension tensor appearing in equation (78) and (79) is calculated once between each two of the three available phases in the case of liquid drop in contact with the solid wall. For instance, for a fluid particle \((i)\) located in the liquid phase \((k)\), two separate tensors would be calculated; one between liquid and gas phases and the other between the liquid and the solid phases. Afterwards, these two calculated tensors would be added together to form the total surface tension tensor for that particular fluid particle in the form of

\[
\Pi_{i}^{total} = \sum_{\forall \neq k} \Pi_{i}^{k,l} \tag{80}
\]

This summation covers all other phases \((l)\) that are located in the neighborhood of particle \(i\). Here, instead of using equation (80) to superimpose the effects of the three phases, only effect of gas and liquid phases on each other is considered. In other words, only the surface coefficients between the gas and liquid phases are taken into consideration. In this manner only one tensor is calculated for each particle which only depends on the opposing phase. This methodology eliminates the need for calculating the effects of liquid-solid and gas-solid phases, and hence brings the complexity of the problem from having three phases to only two phases.

This method introduces two major problems. The contact angle which was to be obtained from the interactions between the two phases would no longer be calculated accurately. Moreover, the function of \(\nabla C^{k,l}_i\) which appears in equation (79) would seem to be lacking some particles in its neighborhood as by eliminating the particles in the solid phase from these calculations, fluid and gas particles no longer are being affected by the solid boundary. This causes unrealistic increase in shear stress near the triple point which tends to stop fluid particles to reach a desirable equilibrium by constantly circulating them inside each phase. To overcome each of these problems, the proposed procedures by Šikalo et al. [2] and Afkhami et al. [3] are jointly used. Šikalo et al. studied variations of dynamic contact angles in droplet impact using VOF method. In their method, the unit normal vector appearing in the contact line cell is recalculated to match the desirable contact angle at the boundary. As suggested in their studies, this correction would introduce a force per unit length equal to \(f_{cl} = \alpha \cos \theta_D\) which is then applied to the contact line.
in the direction parallel to the wall, with $\theta_D$ being the desired dynamic contact angle. This force is only applied to particles near the contact line using the local calculated color function. In more detail, for those particles which are not near the contact line, surface tension is calculated using equations (71), (74) and (76). Please note that equation (75) is no longer needed as only the liquid and gas phase interaction is considered. For the fluid particles located immediate to the contact line, the unit normal instead of being in the form of $\mathbf{n} = \nabla C / |\nabla C|$, is recalculated using

$$\mathbf{n} = \mathbf{n}_f \cos \theta_D + \mathbf{n}_l \sin \theta_D$$

(81)

where $\mathbf{n}_f$ and $\mathbf{n}_l$ are unit normal vectors perpendicular and parallel to the wall, respectively. $\mathbf{n}$ as shown in equation (81) is applied only to the particles which are inside the droplet. For the rest of the particles forming the contact line (vapor phase), the opposite direction of $\mathbf{n}$ is used. Later this corrected normal is substituted into equation (73) and the rest of calculations as suggested by Šikalo et al. can be continued using the normal distribution of the color function (for calculation of $|\nabla C|$ in equation (73)) this calculated tensor is then inserted into equation (76) which results in the proper surface tension force for particles near the boundary. Algorithm 1 shows the steps used for imposing this condition.

**Algorithm 1** – unit normal correction near the triple point

**for all** $i$ particles involved in calculation of equation (74) **do**

**if** $(y(i) <= 1.2 \times$ initial particle spacing in $y$ direction **and** $(\nabla C_x$ or $\nabla C_y$ of particle $i != 0))$ **then**

**if** (type for $i$ = fluid particle) **then**

$$\mathbf{n}_x = \sin \theta_D$$

$$\mathbf{n}_y = \cos \theta_D$$

**else**

$$\mathbf{n}_x = -\sin \theta_D$$

$$\mathbf{n}_y = -\cos \theta_D$$

**end if**

**end if**

**end for**
As it will be discussed later in the validation section, the sole use of this method would lead to drops which have the right contact angle although the profile of the drop is not well constructed. The results suggest that although the normal angle at the triple point can be corrected by this method, the resulted curvature is still lacking enough accurateness. A reason behind this is the use of the local distribution of the color functions in calculation of $|\nabla C|$. $\nabla C$ calculation, as is evident in equation (74), lacks a complete neighborhood of particles for those particles that are near the contact line. To overcome this problem, a procedure similar to the one used by Afkhami et al. [3] is utilized here. In this method, in an attempt to correct the unbalanced calculation of $\nabla C$ near the contact line, the drop profile is interpolated into the solid boundary using a straight line passing from the position of the triple point with a slope perpendicular to the unit normal that is imposed to the contact line particles ($\vec{n}$ in equation (81)). The tangent of this interpolation line ($tan_{int}$) can hence be calculated from

$$tan_{int} = \tan(\pi - \theta_D)$$

(82)

for the case of a liquid drop located on the left bottom corner of the boundary and $\theta_D$ being the angle that the fluid inside the drop is making with the surface measured inside the drop. In this manner, the identities of the particles located inside the wall boundary are temporarily changed based on their positions. Those particles that fall inside the interpolated drop profile would be treated as fluid particles and those left outside would be given values related to the gas phase. Algorithm 2 shows the steps used for imposing this condition. It is important to make sure that when the type of a particle is changed temporarily to belong to each of the phases, variables including the mass and density of the particle should be reassigned based on the new definition. If equation (66) is used for density calculation, the new density can be easily obtained by modifying the existing density of the ghost boundary particle from

$$\rho_{new} = \rho_{old} \cdot \frac{m_{new}}{m_{old}}$$

(83)

The results obtained using these two procedures are presented in the validation part. These results show a very good convergence to the desirable contact angle while keeping the rate of shear stress near the contact line at lower values.
Algorithm 2 – drop profile interpolation into boundary

for all $i/j\,^1$ particles in equation (71) do

if (particle $i/j$ belong to the wall) then

if ($\tan_{int} > 0$) then

if $y(i/j) > \tan_{int} \times x(i/j) - \tan_{int} \times x(contact\ line)$ then

temporary type for $i/j = fluid$ particle

else

temporary type for $i/j = gas$ particle

end if

else

if $y(i/j) < \tan_{int} \times x(i/j) - \tan_{int} \times x(contact\ line)$ then

temporary type for $i/j = fluid$ particle

else

temporary type for $i/j = gas$ particle

end if

end if

end if

end for

\[ ^1 i/j = i\ or\ j \]
4.2. Validation

4.2.1. Oscillating Rod Test

Before validating the contact angle implementation methods, the multiphase model used is validated for investigating robustness and accuracy of the flow solver. The circular liquid drop oscillation test with finite surface tension is performed. A drop with radius of $R = 0.1875$ is located inside a 1x1 rectangular fluid domain. Both fluids have similar densities of $\rho_1 = \rho_2 = 1$ and viscosities of $\mu_1 = \mu_2 = 5 \times 10^{-2}$. The surface tension at the interface is $\alpha = 1$. Due to symmetry, only one fourth of the domain is modeled and no slip boundary condition is imposed on the walls. The computational domain is decomposed into 900 particles with a constant time step of $10^{-4}$. The drop is initially left to reach equilibrium. Then, a divergence free initial velocity is assigned to all the particles located inside the drop, defined by

$$
V_x = V_0 \frac{x}{r_0} \left(1 - \frac{y^2}{r_0^2}\right) \exp\left(-\frac{r}{r_0}\right)
$$

$$
V_y = V_0 \frac{y}{r_0} \left(1 - \frac{x^2}{r_0^2}\right) \exp\left(-\frac{r}{r_0}\right)
$$

(84)

$V_0$ and $r_0$ are constants chosen to be 10 and 0.05, respectively. $x$ and $y$ are the horizontal and vertical distance of each particle from the center of the drop and $r$ is defined as $\sqrt{x^2 + y^2}$. figure 4.1 shows variation of the center of mass position of the quarter drop. The calculated amplitude and period of the oscillation are found to be 0.012 and 0.37 which are in good agreement with the previously reported results of this specific oscillation test case (reported amplitudes from 0.012 to 0.015) [40, 1].

![Figure 4.1. Quarter drop center of mass position versus time](image-url)
4.2.2. Stationary drop with contact angle of 90°

For the case of liquid drop sitting stationary on a solid wall, the results of the model of Hu et al., with the methods of unit normal vector correction and the gradient of color function correction are compared against each other.

![Figure 4.2](image_url)

Figure 4.2. Figure on left showing the initial positioning of particles inside the drop. Figure on the right showing the drop at equilibrium with an angle of 90°. The second phase filling the domain outside the drop has not been shown. 833 particles are sweeping the surface of the drop.

As shown in figure 4.2, a quarter circle drop with a radius of $R = 0.25$ is initially placed inside a domain $0.5 \times 0.5$ with initial spacing of particles being $0.5/65$ (a mesh size of $65 \times 65$ and around 32 particles per radius of the drop). The left boundary is considered symmetric to produce a half circle drop placed on boundary with the size of 1.0. In order to decrease the run time by increasing the time steps, both fluids inside and outside of the drop are assigned equal density and viscosity of 1.0 and 0.15 respectively. A constant time step of $\Delta t = 7 \times 10^{-5}$ is chosen. For the model of Hu et al., three surface tension coefficients in the form of $\alpha^{lg} = \alpha^{sg} = \alpha^{sl} = 1.0$ are chosen which according to Young-Laplace equation, give a stationary contact angle of 90°. The same value of $\alpha^{lg} = 1.0$ is chosen for the case where only one surface tension coefficient is used. For this case, unit normal near the contact line are corrected corresponding to a contact angle of $\theta_{p} = 90°$. Drop profile is also interpolated into the boundary as a vertical line for $\nabla C$ correction. All simulations are run till drops reach their equilibrium.
Figure 4.3. Contact angle deviations from 90° for a half circle drop left to reach its equilibrium using three surface tension coefficients of $\alpha^{dg} = \alpha^{g} = \alpha^{st} = 1.0$

Figure 4.4. Contact angle deviations from 90° for a half circle drop left to reach its equilibrium using one surface tension coefficient of $\alpha^{dg} = 1.0$ along with unit normal and $\nabla C$ correction

figure 4.3 and figure 4.4 show the deviation of the measured value of the contact angle from the desired value of 90° versus time. This contact angle is measured as the angle that the drop surface particle located at nearly two initial spacing from the boundary wall makes with the drop particle located immediate to the triple point. Figure 4.3 is obtained by using three surface coefficients while figure 4.4 shows results using only one surface tension coefficient along with normal and $\nabla C$ correction. Comparing these two figures shows that by adjusting the normal in the boundary particles, the resulting value of the contact angle would be much closer to the
desired value. The trend of variation also suggests that the correction used for $\nabla C$ is also relaxing the particles’ movements and variations near the boundary. This effect could have been expected as by interpolating drop profile in the boundary, fluid particles from the point of calculating $\nabla C$ only see a smooth continuous surface for the drop rather than a discontinued drop profile at the triple point.

Figure 4.5 and figure 4.6 show the variations of total kinetic energy of the particles that are located inside the droplet versus time. Figure 4.5 is for the case which utilizes three surface tension coefficients and figure 4.6 describes the case using only one surface tension coefficient with corrected normal and $\nabla C$.

These figures are also in agreement with previous discussions as they show that the correction of $\nabla C$ would avoid unnecessary movements of particles near the triple point and would eliminate fluctuations in particle positions after equilibrium has been reached.

Figure 4.5. Total kinetic energy of all particles located inside quarter of the drop, using three surface tension coefficients of $\alpha^{lg} = \alpha^{eg} = \alpha^{ri} = 1.0$
Figure 4.6. Total kinetic energy of all particles located inside quarter of the drop, using one surface tension coefficient of $\alpha^H = 1.0$ along with unit normal and $\nabla C$ correction

To support the mentioned discussions, the average shear rate ($\partial u/\partial y$) near the boundary is plotted for each case in figure 4.7 and figure 4.8. These plots show a huge difference between resulting shear values as the maximum average shear in the case using three surface tension coefficients is 160 times larger than the case using one surface tension coefficient with corrections. It may be noticed than in the first case (figure 4.7), the maximum shear rate has occurred near the position of the triple point, which is in analogy with the reports of Afkhami et al. In the second case (figure 4.8), shear rate values are nearly zero (compared to the first case) as the triple point in this case has been almost removed and substituted with a continuous surface profile. In other words, from the point of $\nabla C$ calculation, there are no added effects for the triple point. Since the surface is treated as a uniform profile no extra surface tension stress is introduced. Meanwhile the corrected normal at the triple point based on equation (81) as was discussed before introduces a force per unit length equal to $f_{cl} = \alpha \cos \theta_D$. Here since $\theta_D = 90^\circ$, this force is also zero and hence the correction of the normal vector also does not introduce any shear stress near the contact line. As is seen in the next sections, as $\theta_D$ varies, the created nonzero force along with effects of $\nabla C$ would exceed this shear force which would consequently result in better movement of the triple point in forming the desirable contact angle.
Figure 4.7. Variations of average shear rate along the solid boundary, starting from the centre of the liquid drop ("0" on the x axis above) to the boundary wall on the right ("1" on the x axis above), using three surface tension coefficients of $\alpha_{lg} = \alpha_{rg} = \alpha_{lg} = 1.0$

Figure 4.8. Variations of average shear rate along the solid boundary, starting from the centre of the liquid drop ("0" on the x axis above) to the boundary wall on the right ("1" on the x axis above), using one surface tension coefficient of $\alpha_{lg} = 1.0$ along with unit normal and $\nabla C$ correction

The method used here, next to improving the behavior of the contact line at the triple point, seems to also be contributing to a better reconstruction of the curvature of the drop profile away from the triple point. To investigate this, the deviations of unit normal vectors on the surface from their exact values are compared.

For this purpose, the maximum angle (in degrees) between the calculated normal and the exact normal at each particle near the surface is measured at each time and is shown in figure 4.9. The unit normal vectors to the surface in both methods is calculated from $\hat{n} = \nabla C/|\nabla C|$ while the exact values for the direction of the normal vectors are obtained from $\tan(y/x)$, with $x$ and $y$...
being the Cartesian location of each particle. Since the normal vectors near the triple point are being replaced in the correction method with their exact values, these points have been eliminated from comparison for both cases, for better focusing on the drop profile estimation away from the contact line.

Figure 4.9 shows again that utilizing the correction methods not only brings the unit normal vectors closer to their exact directional values, but also reduces the amount of fluctuations and variations in unit normal vectors all over the solution domain.

![Figure 4.9. Maximum deviation of unit normal vectors near the interface and away from the triple point. The black line with unfilled circles shows the case with three surface coefficients while the red line with unfilled triangles is related to the case of one surface tension coefficient with correction methods](image)

4.2.3. Drops in equilibrium

In this section, a drop is initially positioned in the domain with properties similar to those mentioned in section 4.2.1. This drop is making an angle of 90° with the wall. For studying drop response to other contact angles, the properties of drop are suddenly changed to match a contact angle value of 60°. This means that for the method using 3 separate phases with 3 surface tension coefficients, the values of these coefficients are changed to $\alpha^{lg} = \alpha^{stl} = 1.0$ and $\alpha^{sl} = 0.5$. For the correction method presented before, the surface tension between gas and liquid is chosen to be $\alpha^{lg} = 1.0$ and the value of $\theta_D = 60^\circ$ is used for normal corrections. Therefore $\tan_{int} = \tan(120^\circ)$ should be used for drop interpolation in to the boundary (drop is initially positioned
as shown in figure 4.2). All tests are performed by positioning particles with 0.5/65 space between them. In this case, since the contact line is moving, using proper treatment of the moving contact line with a slip model would be useful. For the moment, instead, cases have been tested with both no slip and free slip boundary conditions. Free slip boundary condition as used by Hu et al. [1] can be acceptable in producing accurate contact angles.

Figure 4.10 shows the variation of resulting contact angles for the cases where a no slip boundary condition is imposed on the solid wall. As is apparent, the angle initially starts from 90° and eventually converges to the value of 60°. This figure also indicates that a more accurate contact angle can be obtained when proper normal and $\nabla C$ corrections are employed.

![Graph showing contact angle deviations from 60°](image)

As is shown in figure 4.12, similar to the results seen in 4.2.1, imposing the correction methods relaxes the shear stresses near the triple point at the time of equilibrium. This outcome improves the convergence behavior of the solution by making the resulting equilibrium more stable.

Figure 4.13 shows drop’s spread factor (D/D0) versus time. This figure clearly demonstrates that the contact line would experience larger movements from its initial position when it is subjected to the correction method. This result can also be backed up by the plotted average shear rate at initial times of drop evolution in figure 4.11. By comparing figure 4.11 with figure 4.12, it can be observed that the shear stress near the contact line in the correction method would be larger in initial stages of drop evolution and it would eventually be more relaxed at the time of equilibrium. Since this initial shear is larger compared to initial shear produced from original 3-
phase method, the contact line in the correction method experiences larger movements. It should also be noted that by using the correction method, shear stress increases at the beginning and decrease as equilibrium arrives while in the original 3-phase method, shear remains almost the same throughout the evolution.

Figure 4.11. Variations of average shear rate at initial stages of drop’s evolution (averaged near time=0.25) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). The red dashed line shows the case with three surface tension coefficients of $\alpha^{lg} = \alpha^{sg} = 1.0$ and $\alpha^{sl} = 0.5$. The black solid line is showing results for the case with $\alpha^{lg} = 1.0$ and normal and $\nabla C$ corrections.

Figure 4.12. Variations of average shear rate at equilibrium (averaged near time=4.4) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop (“0” on the x axis above) to the boundary wall on the right (“1” on the x axis above). The red dashed line shows the case with three surface tension coefficients of $\alpha^{lg} = \alpha^{sg} = 1.0$ and $\alpha^{sl} = 0.5$. The black solid line is showing results for the case with $\alpha^{lg} = 1.0$ and normal and $\nabla C$ corrections.
Figure 4.13. Spread factor of the drop (instantaneous diameter of drop divided by initial drop diameter). The green line with unfilled triangles is showing results for the case with $\alpha^{lg} = 1.0$ and normal and $\nabla C$ corrections where free slip condition is imposed on the boundary. The blue line with unfilled circles is also related to the same case with the difference of having a no slip boundary condition. The dashed red line demonstrates results of the three phase case with $\alpha^{lg} = \alpha^{st} = 1.0$ and $\alpha^{sl} = 0$ where a free slip boundary condition is imposed. The black solid line is also related to the same case with the difference of having a no slip boundary condition.

Figure 4.14 and figure 4.15 show the results related to the case with a free slip boundary condition. Results are still in an acceptable range and also follow the discussions provided before, although it seems that the no slip boundary condition was more accurate in capturing the desired contact angle. It can also be pointed out that the use of free slip on the boundary has made the resulting equilibrium less stable compared to no slip condition, which was predictable.

Figure 4.14. Contact angle deviations from 60° when a free slip boundary condition is imposed. The red dashed line shows the case with three surface tension coefficients of $\alpha^{lg} = \alpha^{st} = 1.0$ and $\alpha^{sl} = 0.5$. The black solid line is showing results for the case with $\alpha^{lg} = 1.0$ and normal and $\nabla C$ corrections.
4.2.4. Convergence test

To study the dependence of the resulting contact angle on the mesh resolution, the following study is conducted. The same drop with the properties mentioned in previous sections is initially place on the wall while making an angle of 90°. Properties of the drop are then suddenly changed to match those of a drop with a contact angle of 60°.

Only the correction method is tested here, hence the surface tension between gas and liquid is chosen to be $\alpha^{lg} = 1.0$ and the value of $\theta_D = 60°$ is used for normal corrections. Therefore $\tan(int) = \tan(120°)$ should be used for drop interpolation in to the boundary (drop is initially positioned as shown in figure 4.2). A no slip boundary condition is also imposed on the lower wall. This test case is repeated for different particle positioning of 0.5/45, 0.5/65, 0.5/85, and 0.5/105.

Figure 4.16 demonstrates shear rate on the boundary at equilibrium. As also captured in studies of Afkhami et al. [3], by refining the resolution, the shear rate near the contact line tends to diverge. Regardless of this increase in shear rate, except for the extremely coarse mesh of 0.5/45, on other mesh resolutions, good convergence is observed for spread factor both during the
spreading and at the final equilibrium state (figure 4.18). Contact angles for different resolutions as demonstrated in figure 4.19 converge to approximately unique value.

Figure 4.16. Variations of average shear rate at equilibrium (averaged near time=4.4) along the solid boundary with a no slip boundary condition; starting from the centre of the liquid drop ("0" on the x axis above) to the boundary wall on the right ("1" on the x axis above). The red line with unfilled triangles shows the case with the resolution of 0.5/105. The green solid line is related to the resolution of 0.5/85. The blue solid line shows results of the 0.5/65 case while the black line with unfilled circles shows the 0.5/45 case. In all cases, $\alpha = 1.0$ and normal and $\nabla\cdot$ corrections are used.

Figure 4.17. Total kinetic energy of all particles located inside the quarter of drop, using surface tension coefficient of $\alpha = 1.0$ along with unit normal and $\nabla\cdot$ corrections. Plotted solid lines with colors of red, green, blue, and black represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively.
Figure 4.18. Spread factor of the drop (instantaneous diameter of drop divided by initial drop diameter) for the case with $\alpha^{ld} = 1.0$ and normal and $\nabla C$ corrections. Plotted data with red line with unfilled triangles, green solid line, blue solid line, and black line with unfilled circles represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively.

Figure 4.19. Contact angle deviations from 60° with a no slip boundary condition for various resolutions. Plotted solid lines with colors of red, green, blue, and black represent cases with resolutions of 0.5/105, 0.5/85, 0.5/65, and 0.5/45 respectively.
Chapter 5
Case study: 2-D droplet impact

In this chapter, the primary effects of the mentioned model on a moving contact line are studied. The test cases chosen here are 2-D drops impacting on a solid surface. Results are then benchmarked against some available experiments and an analytical model developed in the next section. This analytical model is established specifically in 2-D Cartesian coordinates for getting better estimates of the robustness of the 2-D droplet impact model.

5.1. Analytical 2-D Cartesian model

Since the models presented later in this study are two dimensional Cartesian, it is convenient to develop a simplified analytical estimate of drop impact similar to the work of Pasandideh-Fard et al. [52] in order to have results benchmarked against. Figure 5.1 shows a cylindrical infinite fluid element having an initial diameter of $D_0$ impacting on a surface at a velocity of $V_0$ and forming a fluid film with a thickness of $h$, shown in gray, expanding to the sides with a velocity of $V_x$.

The initial kinetic and surface energies of the cylinder before impact are

$$KE_1 = \frac{1}{2} \rho (volume) V_0^2 = \frac{1}{2} \rho \frac{\pi D_0^2}{4} L V_0^2 = \frac{1}{8} \rho \pi D_0^2 V_0^2 L$$

$$SE_1 = \pi D_0 L \alpha$$

(85) (86)

Figure 5.1 2D Cartesian drop impacting on a surface and generating a thin film of liquid beneath
where \( \alpha \) is the surface tension coefficient. After impact, when the drop is at maximum spread, the kinetic energy would be zero while the surface energy is in the form of

\[
SE_2 = D_{max} L \alpha(1 - \cos(\theta))
\]  

(87)

where \( \theta \) is the contact angle. The amount of energy lost in the deformation of drop due to viscosity is \([4]\)

\[
W = \int_0^{t_c} \int_{\Omega} \phi \ d\Omega \ dt \approx \phi \Omega t_c
\]

(88)

where \( \Omega \) is the volume of the viscous layer of fluid, \( t_c \) is the time taken for the drop to reach its maximum spread length \( D_{max} \), and \( \phi \) is the viscous dissipation function estimated as

\[
\phi \sim \mu \left( \frac{\partial u}{\partial y} \right)^2
\]

(89)

Chandra et al. \([4]\) suggested \( \phi \sim \mu (V_0/L^*)^2 \) with \( L^* \) being a characteristic length in the direction perpendicular to the wall which is chosen to be equal to the splat thickness, \( h \) (see figure 5.1). Pasandideh-Fard et al. \([52]\) reported this assumption to overestimate \( D_{max} \) and suggested using the boundary layer thickness, \( \delta \), at the solid-liquid interface as an alternative. They have also shown that this thickness remains almost constant during droplet spreading and therefore assumed \( \delta \) to be equal to the boundary layer thickness at stagnation point. Mao et al. \([53]\) argued that depending on the initial conditions, \( \delta \) could be either smaller or larger than \( h \). By assuming a linear velocity variation inside the viscous layer \( \delta \), viscous dissipation can be calculated from equation (89). \( \Omega \) is also calculated based on the viscous layer thickness which would be the smaller of \( \delta \) and \( h \). All three approaches are modified to produce 2D results which are later compared in table 1. Here, only the derivation of formulations based on the approach of Mao et al. is discussed.

From potential flow theory, for a 2D stagnation point, the stream function is \( \psi = axy \) and therefore, \( V_y = -\partial \psi / \partial x = -ay \). Just before impact, the velocity of the fluid at the center of the drop would be \( V_0 \) at \( y = D_0/2 \) resulting in \( a = 2V_0/D_0 \). Using similarity solution, the two dimensional boundary layer thickness can be obtained from
\[
\delta = \frac{2.4x}{\sqrt{\frac{\rho V_x x}{\mu}}} = \frac{2.4}{\sqrt{\frac{a}{v}}} = \frac{2.4}{\sqrt{\frac{2V_0}{vD_0}}} = \frac{2.4}{\sqrt{2} \sqrt{Re}}
\]  
(90)

Spread time \((t_c)\), the time takes for the drop to reach is maximum expansion, can be obtained from conservation of mass between the droplet and expanding film by calculating the flow passing through the contact area, \(d\), as shown in figure 5.1. In this case, \(\rho V_0 d L = 2 \rho V_x h L \Rightarrow V_x/V_0 = d/2h\). Film thickness, \(h\), after impact is \(\pi D_0^2 L/4 = D_{max} L h\) which results in \(h = \pi D_0^2 /4D_{max}\). Since contact area varies between 0 and \(D_0\), it can be estimated by \(D_0^2 /2\). Plugging back into \(V_x\), the spread velocity is \(V_x = V_0 D_{max}/\pi D_0\).

The film spreading rate is \(dD/dt = 2V_x\). By substituting \(V_x\) and integration over time, \(D/D_{max} = 2t^*/\pi\) where \(t^* = V_0 t/D_0\). The time required to reach maximum spread can be obtained when \(D = D_{max}\); hence, \(t^*_c = \pi/2\) resulting in \(t_c = \pi D_0/2V_0\). For the case of \(h < \delta\), substituting \(\Omega = D_{max} L h\), \(\phi = \mu(2V_x/h)^2\), and \(t_c\) into equation (88) gives

\[
\frac{W}{\alpha D_{max} L} \approx 8 \frac{We}{Re} \left(\frac{D_{max}}{D_0}\right)^3
\]

(91)

and for the case of \(h > \delta\), substituting \(\Omega = D_{max} L \delta\), \(\phi = \mu(hV_x/\delta(h - \delta/2))^2\), and \(t_c\) into equation (88) gives

\[
\frac{W}{\alpha D_{max} L} \approx \frac{2.4 \pi^2}{16 \sqrt{2}} \frac{D_0^2}{D_{max}^2} \frac{1}{\sqrt{Re}} + \frac{1}{4} \frac{2.4^3}{\sqrt{8}} \frac{1}{\sqrt{Re^3}} - \frac{\pi}{8} \frac{2.4^2 D_0}{D_{max} \sqrt{Re}} + \frac{1}{8}
\]

(92)

Plugging equations (85), (86), (87), and (91) or (92) into energy conservation equation, \(KE_1 + SE_1 = SE_2 + W\), results in
\[
D_{\text{max}} \leq \frac{\pi}{8} We + \pi \quad (1 - \cos(\theta)) + \frac{\pi We}{Re \left( \frac{D_{\text{max}}}{D_0} \right)^3} \\
D_{\text{max}} = \frac{\pi}{8} We + \pi \quad \frac{2.4\pi^2}{16\sqrt{2}} \frac{D_0^2}{D_{\text{max}}^2} \frac{1}{\sqrt{Re}} + \frac{1}{4} \frac{2.4^3}{\sqrt{8}} \frac{1}{\sqrt{Re^3}} = \frac{\pi}{8} \frac{2.4^2 D_0}{D_{\text{max}}} \frac{1}{Re} 
\]

Table 1 Comparison of measured values of spread factor with predictions, showing results from axisymmetric model of Pasandideh-Fard et al. and 2D Cartesian results obtained using assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93)

<table>
<thead>
<tr>
<th>(Re)</th>
<th>(We)</th>
<th>(\theta_D)</th>
<th>(V_0)</th>
<th>(D_0)</th>
<th>(D_{\text{max}}/D_0)</th>
<th>(D_{\text{max}}/D_0)</th>
<th>(D_{\text{max}}/D_0)</th>
<th>(D_{\text{max}}/D_0)</th>
<th>Measured</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>802</td>
<td>1.693</td>
<td>4.1</td>
<td>0.00245</td>
<td>1.61</td>
<td>4.34</td>
<td>6.41</td>
<td>2.78</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>802</td>
<td>0.297</td>
<td>4.1</td>
<td>0.00245</td>
<td>1.62</td>
<td>4.41</td>
<td>6.48</td>
<td>2.79</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>51</td>
<td>1.693</td>
<td>1.04</td>
<td>0.00245</td>
<td>1.22</td>
<td>2.27</td>
<td>3.22</td>
<td>1.99</td>
<td>1.5</td>
<td>[2]</td>
</tr>
<tr>
<td>27</td>
<td>51</td>
<td>0.297</td>
<td>1.04</td>
<td>0.00245</td>
<td>1.26</td>
<td>2.54</td>
<td>3.49</td>
<td>2.03</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>93</td>
<td>1.693</td>
<td>1.41</td>
<td>0.00245</td>
<td>1.27</td>
<td>2.56</td>
<td>3.71</td>
<td>2.13</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>2112</td>
<td>27</td>
<td>1.92</td>
<td>1.00</td>
<td>2.05</td>
<td>2.47</td>
<td>7.29</td>
<td>9.40</td>
<td>4.73</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>2112</td>
<td>27</td>
<td>1.92</td>
<td>1.00</td>
<td>2.02</td>
<td>2.47</td>
<td>7.29</td>
<td>9.40</td>
<td>4.73</td>
<td>2.16</td>
<td>[52]</td>
</tr>
<tr>
<td>2112</td>
<td>28</td>
<td>1.92</td>
<td>1.00</td>
<td>2.07</td>
<td>2.49</td>
<td>7.42</td>
<td>9.62</td>
<td>4.75</td>
<td>2.62</td>
<td></td>
</tr>
<tr>
<td>2084</td>
<td>59</td>
<td>1.937</td>
<td>2.61</td>
<td>0.616</td>
<td>2.77</td>
<td>10.30</td>
<td>14.80</td>
<td>5.00</td>
<td>2.65</td>
<td></td>
</tr>
<tr>
<td>3298</td>
<td>118</td>
<td>1.937</td>
<td>3.29</td>
<td>0.776</td>
<td>3.25</td>
<td>15.18</td>
<td>22.77</td>
<td>5.58</td>
<td>3.18</td>
<td>[54]</td>
</tr>
<tr>
<td>$Re$</td>
<td>$We$</td>
<td>$\theta_D$</td>
<td>$V_0$</td>
<td>$D_0$</td>
<td>$D_{max}/D_0$</td>
<td>$D_{max}/D_0$</td>
<td>$D_{max}/D_0$</td>
<td>$D_{max}/D_0$</td>
<td>Measured</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
<td>------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>4258</td>
<td>171</td>
<td>1.937</td>
<td>3.71</td>
<td>0.888</td>
<td>3.55</td>
<td>18.58</td>
<td>28.21</td>
<td>5.91</td>
<td>3.45</td>
<td></td>
</tr>
<tr>
<td>5057</td>
<td>219</td>
<td>1.937</td>
<td>4.00</td>
<td>0.977</td>
<td>3.75</td>
<td>21.18</td>
<td>32.32</td>
<td>6.13</td>
<td>3.79</td>
<td></td>
</tr>
<tr>
<td>5833</td>
<td>271</td>
<td>1.937</td>
<td>4.28</td>
<td>1.053</td>
<td>3.94</td>
<td>23.60</td>
<td>36.10</td>
<td>6.31</td>
<td>3.91</td>
<td></td>
</tr>
<tr>
<td>2084</td>
<td>59</td>
<td>1.082</td>
<td>2.61</td>
<td>0.616</td>
<td>3.24</td>
<td>15.24</td>
<td>22.52</td>
<td>5.29</td>
<td>3.15</td>
<td></td>
</tr>
<tr>
<td>3298</td>
<td>118</td>
<td>1.082</td>
<td>3.29</td>
<td>0.776</td>
<td>3.64</td>
<td>20.34</td>
<td>30.51</td>
<td>5.76</td>
<td>3.56</td>
<td></td>
</tr>
<tr>
<td>4258</td>
<td>171</td>
<td>1.082</td>
<td>3.71</td>
<td>0.888</td>
<td>3.89</td>
<td>23.78</td>
<td>35.75</td>
<td>6.05</td>
<td>3.82</td>
<td></td>
</tr>
<tr>
<td>5057</td>
<td>219</td>
<td>1.082</td>
<td>4.00</td>
<td>0.977</td>
<td>4.08</td>
<td>26.36</td>
<td>39.66</td>
<td>6.24</td>
<td>4.1</td>
<td></td>
</tr>
<tr>
<td>5833</td>
<td>271</td>
<td>1.082</td>
<td>4.28</td>
<td>1.053</td>
<td>4.23</td>
<td>28.72</td>
<td>43.19</td>
<td>6.41</td>
<td>4.24</td>
<td></td>
</tr>
<tr>
<td>2084</td>
<td>59</td>
<td>0.471</td>
<td>2.61</td>
<td>0.616</td>
<td>3.59</td>
<td>20.16</td>
<td>28.62</td>
<td>5.44</td>
<td>3.47</td>
<td></td>
</tr>
<tr>
<td>3298</td>
<td>118</td>
<td>0.471</td>
<td>3.29</td>
<td>0.776</td>
<td>3.90</td>
<td>24.60</td>
<td>35.70</td>
<td>5.85</td>
<td>4.07</td>
<td></td>
</tr>
<tr>
<td>4258</td>
<td>171</td>
<td>0.471</td>
<td>3.71</td>
<td>0.888</td>
<td>4.11</td>
<td>27.73</td>
<td>40.50</td>
<td>6.12</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>5057</td>
<td>219</td>
<td>0.471</td>
<td>4.00</td>
<td>0.977</td>
<td>4.27</td>
<td>30.12</td>
<td>44.13</td>
<td>6.30</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>5833</td>
<td>271</td>
<td>0.471</td>
<td>4.28</td>
<td>1.053</td>
<td>4.41</td>
<td>32.29</td>
<td>47.40</td>
<td>6.46</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>26</td>
<td>2.443</td>
<td>0.50</td>
<td>2.99</td>
<td>1.75</td>
<td>3.91</td>
<td>5.48</td>
<td>3.06</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>427</td>
<td>102</td>
<td>2.443</td>
<td>1.00</td>
<td>2.99</td>
<td>2.13</td>
<td>6.82</td>
<td>10.25</td>
<td>3.72</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>641</td>
<td>230</td>
<td>2.443</td>
<td>1.50</td>
<td>2.99</td>
<td>2.41</td>
<td>9.186</td>
<td>13.86</td>
<td>4.10</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>854</td>
<td>410</td>
<td>2.443</td>
<td>2.00</td>
<td>2.99</td>
<td>2.62</td>
<td>11.13</td>
<td>16.74</td>
<td>4.37</td>
<td>2.7</td>
<td></td>
</tr>
</tbody>
</table>

[Sources: 52, 55]
In table 1, axisymmetric model of Pasandideh-Fard et al. and 2D Cartesian results obtained using assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93) are presented. As compared by experimental results, it can be observed the 2D analytical results obtained from the assumptions of (a) Pasandideh-Fard et al. and (b) Chandra et al. are producing a considerable amount of error. The axisymmetric model of Pasandideh-Fard et al. seems to be producing close results to experiments. 2D results obtained from equation (93) would be standing in second place. Since the experimental results here are from 3D droplets impacting the surface, in the next section, by the means of the SPH solver, a 2D drop impact case will be studied and compared against these 2D models.

### 5.2. 2D drop impact with Constant Contact Angles

In order to have a more quantitative comparison, a two-dimensional test case is studied in which a constant contact angle is imposed during the impact of a water droplet with a radius of 250 μm. The chosen values of constant contact angles for each case are 50º, 70º, 90º, 100º, 110º, 130º,
145°, 160°, and 175°. Droplet is impacting the surface from a distance of 375 μm at a velocity of 1 m/s under gravitational force of 9.8 m/s². The calculated Reynolds and Weber numbers are 440 and 6.86, respectively. The computational domain is a square with sides of 3×375 μm filled with 10,000 particles. 776 particles sweep the surface of the drop (approximately 23 particles per radius) and the rest of particles form the surrounding air. Figure 5.2 shows only particles inside the drop before impact. As demonstrated, only half of the drop is simulated here by taking the vertical y-axis as the symmetry line. Results of these impacts have been shown and compared in the following figures. For imposed angles larger than or equal to 90°, drops experience an expansion on the solid surface and after reaching their maximum expansion diameter, start recoiling. For angles smaller than 90° (test cases of 70° and 50°) drops imping from their centre shortly after reaching their maximum diameter. Here, for the purpose of comparing the maximum spread diameters with the results of analytical models, only expansions till reaching the maximum diameters are of interest in order to be compared to calculated values of equation (93).

![Figure 5.2](image.png)

**Figure 5.2.** Impacting drop on a solid surface shown at initial rest position before the start of impact.

Non-dimensional diameters (D/D₀) of these impact tests have been plotted in figure 5.3 versus non-dimensional time (4μt/ρD₀²). Results have been only shown to the point where drop reaches its maximum expansion position. As is evident in this figure, when angles smaller than 90° are imposed, drops tend to act more hydrophilic and expand more on the surface showing a more
wetting behavior. As the contact angle is increased to larger values above 90°, drops act more hydrophobic and show less wetting behaviors. Hence, for larger values of contact angles, the amount of drop’s expansion on the surface decreases noticeably (as shown in figure 5.3 and figure 5.4).

The maximum spread diameter of the drop can be compared against analytical results presented in previous section. This comparison is shown in figure 5.5. Red dots on this figure demonstrate the non-dimensional maximum diameter of impacted drops gained from the SPH solver for different contact angles (results shown in figure 5.3 and figure 5.4). Other lines in figure 5.5 are plotted according to the formulations behind table 1. As shown in figure, other charts are related to the axisymmetric model of Pasandideh-Fard et al., and 2D Cartesian results obtained from assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93).
Figure 5.3. Non-dimensional diameter ($D/D_0$) of spreading drops during impact versus non-dimensional time ($4\mu t/\rho D_0^2$) for various constant contact angles.
Figure 5.4. Impacted drops shown at their maximum expanded diameter for various constant contact angles imposed during impact.
Figure 5.5. Maximum non-dimensional spread diameter for various contact angles, obtained from different models: axisymmetric model of Pasandideh-Fard et al., 2D Cartesian results obtained from assumptions of (a) Pasandideh-Fard et al., (b) Chandra et al., and (c) equation (93).

For results related to equation (93), both parts of this equation for both conditions of $h < \delta$ and $h > \delta$ are plotted in figure 5.5; although the test cases performed here for the impact fit to the condition of $h > \delta$.

Results show that spread diameters obtain from the SPH model here are also closer to the axisymmetric model of Pasandideh-Fard et al. and 2D results obtained from equation (93). For smaller contact angles, impact results seem to be converging to values obtained from equation (93) using the condition of $h > \delta$, which is the condition matching the test cases here. For larger contact angles, impact test results tend to be converging to the values obtained from equation (93) using the condition of $h < \delta$. 
Chapter 6
SPH on CUDA

6.1. GPU architecture

CUDA, introduced in 2006, is a parallel computing platform that enables programming on Graphic Processing Units or GPUs. CUDA has so far been used in many research areas.

Each GPU consists of a number of threads. These threads are each a processing unit that can perform calculations parallel to each other. The GPU architecture demands that when a program is running on the GPU device, these processing threads should be formed into predefined structures. Depending on the choice of user, these structures can be one, two or three dimensional. These different arrangements would affect how each thread is assigned its identification number. A group of these threads are gathered together as a GPU block (figure 6.1). The number of allowed threads per block may vary depending on the GPU device. These blocks are then gathered to form GPU grids. In other words, any program launched on GPU would launch a grid formed from GPU blocks, while each of these blocks would have a number of threads inside.

Each of these threads would then have access to two separate memories called local memory and registers. By default, all variables that are defined inside a thread are stored in registers. Register memory is extremely fast and efficient; therefore there is always the challenge of keeping the data in this memory. The amount of registers assigned to each thread depends on the type of GPU device as well as the number of threads launched. NVIDIA has provided simple tools for calculation of the amount of available registers (see [57]). The programmer has also the advantage of checking the number of registers used by the launched program. When registers are full, data is automatically transferred to the local memory. Local memory is slower though the read write process from it is automatically coalesced meaning there is no conflict slowing the data transfer.

Then there exists a shared memory, which is a memory shared between all the threads inside a block. All threads in a block have access to shared memory and can share and exchange
information on this space. Threads in other blocks do not have access to this specific shared memory. Shared memory is rather fast although they need extra attention in order to make sure the program routines do not conflict in read/write processes (bank conflicts). Avoiding bank conflicts can increase efficiency and speed by far.

Above all is global, constant and texture memories where all threads in any blocks can access them. These three memories are quiet slow although they have the advantage of being shared with all the threads inside the launched grid. Global memory is analogous to RAM memory on CPU. Moreover, data stored on the global, constant and texture memory is accessible by the CPU host. Therefore, different calculated parameters can be transferred to/from GPU by means of these memories. Texture memory has been also optimized for specific uses which are not discussed here (see [58, 59] for samples of SPH works utilizing texture memory structures).

Figure 6.1. A one dimensional GPU grid holding two GPU blocks. Each block, as shown, contains four threads (arranged in two dimensions here).
6.2. SPH implementation on GPU

The SPH method proposed in sections 3.2.3 and 4.1 has been implemented on GPU using CUDA. The solver outline is demonstrated in figure 6.2. Primarily, the CPU device allocates most of necessary variables on both CPU and the global memory of the GPU devices for easier and quicker later access. Constant values related to each particle, i.e. initial position in the domain, viscosity, phase, and mass of each particle are then defined. Then, the main program loop is initiated.

![Flowchart demonstrating SPH-GPU solver](image.png)
Depending on the total number of available particles and variables associated with each of them, there might be size limitations for solving the whole domain entirely on the GPU. This size limitation is more pronounced when dealing with multiphase SPH solvers, especially when CSF model is in use which demands more variables than the normal SPH methods. Therefore, the main domain is divided into smaller subdomains with a predefined size based on the memory capacity of the available GPU.

Each subdomain is transformed into an individual domain by means of ghost particles which on the downside increases the total number of particles and associated computational time. This procedure cannot be avoided since GPU architecture does not support message passing between GPU blocks. Therefore, calculations are obliged to have check points so that all blocks can be synchronized by leaving data on the global memory. Some studies, like Harada et al. [59], have suggested using approximation functions rather than actually using ghost particles, which is not the method adopted here.

6.3. Searching for Neighboring Particles

During the simulations, particles located in the neighborhood of each particle should be defined and occasionally updated. To increase the performance and speed of the flow solver, different neighboring particle search and updating algorithms are benchmarked here.

There are mainly two possible ways of implementation for the neighboring particle search algorithm: one is the Direct Search method in which all particles are searched to point out those who fall in the neighborhood of a specific particle. The other method, suggested by Liu et al. [18], is the Nearest Neighboring Particle Search method (NNSP). In NNSP (shown in the third column in figure 6.2), the computational domain is divided into equal subdomains. In the first step, a tracking list is generated to assign particles to their corresponding subdomain. The search algorithm is then only limited to the neighboring subdomains of the target particle. Here, this approach is used. A larger neighborhood for each particle is chosen. All the particles fallen into this large neighborhood are tracked and added to a tracking list for that specific particle. In the rest of calculations, search for actual neighbors of that particle is only limited to particles in its tracking list. This tracking list needs to be updated after some iteration. The frequency of these
updates depends on the size of the larger neighborhood and maximum expected velocity of the particles inside the domain. Figure 6.3 shows a comparison between these two methods on different platforms. As shown, for a code compiled completely on CPU, runtime is 30 times faster when using NNSP instead of the Direct Search method. By switching from CPU to GPU using CUDA, the NNSP method runtime is boosted by 4 folds. Performance can be even improved by an extra 6 folds when the tracking list generation is also transferred over to the GPU.

The rest of the functions shown in figure 6.2 would calculate density, surface tension, and viscous effects. The calculated values can then be used for updating velocities and positions of each particle using any proper time marching method.

![Figure 6.3. Runtime comparison on CPU and GPU for different neighbor search algorithms. In these tests, 2025 particles filled the domain while multiphase SPH formulations as discussed before were solved. The y-axis shows the number of time steps per second.](image)

### 6.4. Dimension of Kernels

GPU architecture allows performing parallel instructions. Each computing thread can be assigned an individual task which will be performed in parallel with other threads. These processing threads can be dynamically grouped in the form of one, two, or three dimensional...
blocks. All threads inside a block have access to a local shared memory at the same time and have the advantage of being synced together. These blocks when grouped together would form a grid. Blocks inside a grid do not share local memory with each other and only have access to a global memory. There is also no guarantee that all these blocks inside a grid will be executed at the same time.

6.4.1. One particle per multiple threads

In order to check performance dependence on block configuration, two thread arrangements have been tested. In the first algorithm, threads form two dimensional blocks with a size of 32×16. Each of the 32 rows is assigned to a single particle while the 16 thread columns in each row are assigned to a single neighbor of that specific particle. As each particle is chosen to have a maximum number of 64 neighbors, grids should consist of 4 blocks in $y$ direction (16×4). In the $x$ direction, grids have ($total$ $number$ $of$ $particles$) / 32 blocks in order to make sure that each particle is assigned a unique row of 64 threads in total.

6.4.1.1. Atomic Operations

In this type of arrangement, race conditions are more likely to happen. A race condition is when two or more threads are trying to change a single variable at the same time. For instance, when calculating the density of a single particle using equation (66), in worst case scenario, each of the 64 threads in charge of neighbors of that particle would be attempting to add their effects on the density. Race conditions would make the outcome results unreliable as many threads might read the same value of a variable, add their effect on it, and then overwrite the effects of other threads. A solution to this problem is using atomic operations (see [60] for details on atomic operations). These are functions designed to avoid race conditions. Operations performed under atomic operations would be executed in series rather than parallel whenever race condition occurs. Figure 6.4 shows a schematic view of this process.
\[
\rho_i = \sum m_i W_{i,j} = m_i W_{i,\text{neighbour}_1} + m_i W_{i,\text{neighbour}_2} + m_i W_{i,\text{neighbour}_3} + \ldots
\]

Figure 6.4. showing how each processing thread reads (or calculates) the value of smoothing function, then has to add the needed effects on the density using an Atomic operation in order to avoid conflict with other threads.

If atomic operations are used solely on each thread, 16×4 race conditions discussed before would emerge 64 serial operations (if all 64 threads associated with a single particle are running at the same time. This number may vary depending on how GPU decides to execute the blocks).

6.4.1.2. Reduction

Runtime can be improved by using parallel reduction algorithms like the one suggested by Harris [61]. Reduction methods consist of various algorithms with predefined task-assigning patterns to a number of processing threads. These procedures are meant to avoid race conditions without using serialization solutions. At the same time, they prevent most of the threads from being left idle and also try to keep memory read/write patterns efficient. In this way, runtime can be decreased effectively.
6.4.1.2.1. Different reduction methods

For instance, if it is assumed that summation of all members of the following array of numbers is needed to be calculated.

<table>
<thead>
<tr>
<th>Array</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

Without using reduction, this process can be done correctly by two means; one is to assign a single processing thread to do the summation of all the above numbers. This way, no race condition occurs. However, the summation is performed in a serial manner (if all the rest of available processing threads are left idle). The second method is to assign 8 separate threads to these 8 numbers and have them add their value to a destination variable using an atomic add operation. Although more threads are now involved in the calculation, the outcome process would still be a serial summation.

Instead of these methods, reduction procedures can be employed. One of the primary reduction methods available is done by still assigning 8 threads to these 8 numbers,

<table>
<thead>
<tr>
<th>Thread #</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

and then having the threads with an even thread number to perform a single summation of their corresponding array member with the array member related to the thread with an odd id number located immediate on their right:

<table>
<thead>
<tr>
<th>Thread #</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For the next step, only threads located at numbers dividable by 4 would participate as shown below:
And this procedure continues till all numbers are added together. Here the last step would be when only thread numbers dividable by 8 participate which would leave the final summation result in the first array member.

In practice, this array should be allocated on the shared memory so that all the threads located inside each block can have access to the shared data. Since GPU blocks do not share data, if the array was larger to fit in one block, separate kernels can then be launched for performing separate stages of the reduction. In each block, the reduction can be completed till the results generated by other blocks are needed. Then a separate kernel should be launched to complete the results generated by previously launched blocks. Figure 6.5 shows how a sample reduction of an array can be broken into two kernels.
As suggested by Harris [61], the reduction algorithm discussed before can be further improved. The main disadvantage of this algorithm is having many idle threads. All the odd numbered threads in this algorithm are left idle. The resulting improved algorithm would be in the form shown in figure 6.6.

In the improved method, the number of assigned threads is equal to only half of the size of the array (in figure 6.6, only two threads 0-1 are assigned to the block containing the array instead of four threads of 0-1-2-3). This is done in order to avoid having idle threads. In the previous algorithm, odd threads were idle during the whole reduction process.
Array members in this method are also accessed differently. For instance, for thread 0, it adds the value of first and third array members together instead of adding the first and second members of the array. Although this might not be of a significant difference on paper, it has been shown [61] that this method of memory read/write would avoid shared memory bank conflicts. Bank conflict is when multi threads reading/writing process conflicts with each other leading to a more serialized operation and hence longer runtimes (see [62] for more details).

6.4.1.2.2. Reduction method implementation

Here, the reduction methods are only used inside each GPU block. The main objective is to have as many processing threads as possible engaged in the reduction procedure which is performed on the GPU shared memory. Subsequently, in order to save memory space, instead of completing the reduction between GPU blocks by leaving the data on global memory and re-launching the reduction again by a second kernel (as discussed earlier in figure 6.5), atomic operators are used to add calculated values of each block to the destination variables. Figure 6.7 shows how this process is done for calculating density of each particle. Inside each reduction box, the improved reduction algorithm discussed before is used. Then atomic operations would add the results obtained by each block.
This process reduces complexity of the race condition from $O(mn)$ to $O(n)$, with $m$ being the total number of threads per block related to a specific particle and $n$ the total number of active blocks. For instance, for the case of having blocks with dimensions of $32 \times 16$ mentioned earlier in section 0, 16 threads existed in each of the 4 blocks that would add a value to the density of a single particle. By first performing a reduction within each block, only 4 race conditions between the 4 blocks will be left for atomic operations to handle. This is a much smaller number compared to $4 \times 16$ race conditions that might happen if each thread adds its value directly using atomic operations. This suggests that using the reduction algorithms along to reduce the number of atomic operations used can hugely decrease runtime. Different test cases, also confirm this
statement as expected. Hence here, only the second method of using reduction methods is implemented into the SPH solver.

As demonstrated in figure 6.8, by using the right amount of registers memory (see [57]) and reduction algorithms, a 77 times faster runtime is experienced on GPU compared to the corresponding CPU version.

In the next section, a different approach is studied which shows how atomic operations can be avoided for obtaining better performance.

![Figure 6.8. Runtime improvement by compiling more routines on GPU. In these tests, 2025 particles filled the domain while multiphase SPH formulations as discussed before were solved.](image)

## 6.4.2. One particle per thread

Another practical method to avoid atomic operations is to change threads responsibilities by assigning each individual particle to a single thread. In this approach, threads form two dimensional blocks with a size of 32×1. Each of the 32 threads is assigned to a single particle. Loops inside each thread would make sure that all 64 neighbors of each particle are involved in calculations. Grids should consist of 1 block in y direction and \((total\ number\ of\ particles) / 32\)
blocks in $x$ direction. In this approach, only one thread at a time is attempting to write an input into the variables of a specific particle.

Figure 6.9 compares the GPU global memory usage, GPU load, and the time required for 200 iterations on 10,000 particles for this method against the algorithms mentioned before. As shown, avoiding atomic operations can lead to $\sim 17\%$ reduction in computational time.

### 6.5. Memory Management

In search for better maintenance, two different main routines are also compared here. These routines affect the overall configuration of the solver in all subroutines. In the first method, certain variables, like smoothing kernels and their derivatives, are calculated only once and stored on global memory of the GPU device for further referencing. Test cases have shown that for medium size domains, this can decrease runtime by far compared to recalculating values whenever needed since reading some of these values from global memory can be much faster compared to recalculating them each time.

Since data allocation requires considerably large amount of memory, restrictions are enforced on the size of each individual subdomain which can be sent from host to device. This increases the number of ghost particles used which in return raises the number of variables transferred between host and device. The larger transfer rate eventually leads to higher GPU idle times and lower efficiency (see line ○ in figure 6.9(b)).

On the other hand, by calculating variables on demand instead of saving them on global memory after first calculation, larger domain sizes can be handled and better performance is achieved, as shown in figure 6.9(c). This is mainly due to the ability to increase the GPU load (figure 6.9(b)) which makes up for the time elapsed transferring data between the host and device. Referring to figure 6.9(c), it should be noted that by constantly increasing the number of particles, runtimes does not necessarily decrease at a constant rate (see line Δ and □ in figure 6.9(c) at around 4500 particles). This is a side effect of the larger amount of data that needs to be transferred and processed when the number of particles is increased.
By performing the same benchmark tests used in figure 6.3 and figure 6.8, it can be seen that by avoiding atomic operations and having variables recalculated on demand, runtime can be increased up to 120 times when compared to the NNSP CPU version.

Figure 6.9. GPU performance analysis versus the number of particles in each subdomain. (∙) atomic operations used and most variables calculated once and saved for referencing, (∆) no atomic operations used and variables recalculated on demand, and (□) atomic operations used along with reduction algorithms and variables recalculated on demand
Chapter 7
Closure

7.1. Concluding remarks

Overall it was observed that by memory management and efficient task assigning, a GPU executed program, can become more effective. As shown in previous sections, for the multiphase SPH solver developed here, runtime was 120 times faster on the GPU compared to the CPU version of the solver. Keeping the memory free and meanwhile having all processing threads active was shown to reduce transfer rates and increase competency.

The results of the new implementation of contact angle are really promising. The resulting code by the two proposed procedures is more accurate and fluid motion around the triple point is much more stable.

The 2D analytical results were close to the results obtained from experiments and SPH solver. More detailed behaviors of this model can be studied by comparing it to more different test cases.

7.2. Future work

The following are recommendations for the extending the present work:

- A 3D solver can be developed based on the proposed 2D model here. The GPU solver as well as the proposed method for contact angle implementation should be adjusted for better simulation of 3D drops.

- Adaptive particle positioning can be used along this solver to decrease run time by eliminating unwanted fine particle positioning at points not needed.

- The 2D analytical model can be studied in more detail by having it compared to a variety of test cases for different Re, We and contact angles. Investigations should be conducted on the validation of the assumptions made in this model. More results obtained from experiments and SPH solver can be used for this purpose.
- Different slip systems should be fitted into the solver for more relaxation near the triple point.

- A dynamic contact angle calculation model should be implemented like the ones proposed by Afkhami et al. [3], Sikalo et al. [2], and Kistler [63].

- The GPU solver can become more efficient by using other resources like texture memory. The surface tension calculation method of the SPH solver can also be changed to a more preferable model to have fewer complications.
References


March 2004.


[58] A. Kolb and N. Cuntz, "Dynamic Particle Coupling for GPU-based Fluid Simulation".


