Computational Simulation of Freely Falling Water Droplets on Graphics Processing Units

Jeremy Appleyard

PhD Thesis

Cranfield University
School of Engineering

Cranfield University

Jeremy Appleyard

Computational Simulation of Freely Falling Water Droplets on Graphics Processing Units

School of Engineering

PhD Thesis

Academic Year: 2012-2013

Supervisor: Professor Dimitris Drikakis

August 2013

Abstract

This work describes and demonstrates a novel numerical framework suitable for simulating the behaviour of freely falling liquid droplets. The specific case studied is designed such that the properties of the system are similar to those of raindrops falling through air. The study of raindrops is interesting from both an engineering standpoint and from a standpoint of pure curiosity. As a natural phenomenon, rainfall is something which is experienced by everybody, yet its properties are often misunderstood. The primary engineering application is in improving the ability of radar to determine the characteristics of rainfall for meteorological purposes.

The significant original contributions to knowledge within this work come from several areas. The numerical methods used are a unique combination of a high order incompressible implicit large eddy simulation method, a conservative level set method, and a pressure projection method. These methods have all been implemented on a highly parallel GPU architecture, with a resulting performance increase of approximately ten times when a single GPU was compared to a single CPU core.

The water droplets were simulated in a regime not previously studied by three dimensional methods. The results of these simulations confirmed the validity of the numerical model by reproducing several important experimental results. New insight was then gained regarding the behaviour of droplet wakes, an area with little previous research. The results of the test simulations show great promise for future use of the numerical framework developed. While the simulations to-date have been of air-water interactions, there is little reason the model should be constrained to such a system. In theory almost any low speed isothermal interaction of immiscible Newtonain fluids, with length scales of greater than 1mm, could be modeled accurately by these methods.

Acknowledgments

Firstly, I would like to thank Professor Drikakis for the opportunity to work here at Cranfield and for his advice throughout the past years. His constant enthusiasm for the project was reassuring. I would also like to thank the Engineering and Physical Sciences Research Council (EPSRC) for their financial support.

In my time here I was blessed (or perhaps cursed!) to have been in a very large office with some very friendly people. It has always been quite an interesting office to work in, with entertaining conversation daily. While many office-mates have come and gone, Pierre Jameson and Mike Probyn deserve a special mention.

A special thanks to my friends and housemates Jason Barden and Klemontina Gerova. Not only have they helped to keep me motivated through the final stages of my PhD, but also gave me a place to sleep for several months. I am very much looking forward to their wedding.

Outside of the university I'd like to thank friends Emma Cull and Andrew Pay. Emma for her unwavering faith and support, and Andrew for his very comfy sofa (and a bit of faith and support here and there). I'd also like to thank Daniel Hardman, David Plumstead, Ailsa Redhouse, Steffi Sgoda, Daniel Strange and Kevan Mehta, all of whom I have come to rely on for friendship and support. Finally, I'll like to thank Jack Paget, whose feedback helped to fix many small errors in spelling and grammar in the draft version of this work.

Finally I'd like to thank my family. My grandmother Mary, my two brothers Peter and Robert and my parents John and Carole. My parents, especially, have been nice enough to provide me with a place to stay when I needed to be away from university, and have both been enthusiastic about this project, without applying too much parental pressure.

Jeremy Appleyard

Contents

List of Figures

List of Tables

Notation

α	Volume fraction
Δt	Time step
Δx, Δy, Δz	Cell spacing in the x, y and z directions respectively
$\hat{u}$	Non-dimensional flow velocity
κ	Curvature
κ_{lvl}	Level set curvature
κ_{sq}	Least squares curvature
$\tilde{\mathbf{u}}$	Intermediate velocity vector
$\mathbf{A}$	Co-efficient matrix
$\mathbf{b}$	Right hand side vector
$\mathbf{E}$, $\mathbf{F}$	Velocity fluxes in the x and y direction respectively
$\mathbf{f}$	Body forces per unit volume
$\mathbf{N}$	Normal vector field
$\mathbf{n}$	Normal vector
$\mathbf{p}$	Pressure vector
$\mathbf{q}$	Quality factor
$\mathbf{T}$	Stress tensor
$\mathbf{u}$	Velocity vector
$\mathbf{u}_I$	Interface velocity
$\mathbf{u}_{LS}$	Level Set velocity vector

$\mathbf{x}$	Direction vector
μ	Dynamic viscosity
ω_0	Natural frequency
ω_0^R	Natural frequency (Rayleigh)
ϕ	Signed distance
ρ	Density
σ	Surface tension coefficient
τ	Viscous stress tensor
θ	Interpolant
$A_0^{(2,1)}\left(n,m\right)$	Frequency modification factor
A_α	Peak-to-peak axis ratio amplitude
C_t, V_t, G_t, S_t	Time stepping terms
D_0	Equivalent spherical diameter
$H\left(\phi\right)$	Heaviside step function
N_{faces}	Numer of faces
N_S	Number of small cells
U_∞	Free stream velocity
u_{LS}	Level Set velocity
u_t	Tangential velocity
A	Area
f	Frequency
g	Acceleration due to gravity
i	Iterator
L	Characteristic length
n	Current iteration

p	Pressure
r	Droplet radius
s	Wave speed
t	Time
u, v, w	Velocity in the x, y and z directions respectively
V	Volume
x, y, z	Cartesian directions
K	Preconditioning matrix

Dimensionless Numbers

Eö	Eötvös number
Mo	Morton number
Re	Reynolds Number
St	Strouhal Number
We	Weber number

Abbreviations

2D	Two dimensional
3D	Three dimensional
BiCGStab	Bi-Conjugate Gradient Stabilized
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
GFLOPS	Gigaflops (one billion floating point operations) per second.
GFM	Ghost Fluid Method
GPU	Graphics Processing Unit
HOUC	High-Order Upstream Central

1. Introduction

1.1. Computational Fluid Dynamics and Unsteady Flows

The methods used and described in this work are intended to be used to study the dynamics of fluids computationally. This field of is known, rather unsurprisingly, as computational fluid dynamics (usually abbreviated to CFD). In colloquial usage the word *fluid* is often taken to be synonymous with *liquid*, however in science this is not the case. The Oxford English Dictionary defines a fluid as: "a substance that has no fixed shape and yields easily to external pressure; a gas or (especially) a liquid". The subset of fluids this work is concerned with are Newtonian fluids, in which the stress acting on the fluid is proportional to the deformation caused by that stress. Such fluids are common in everyday life and most fluids people encounter, such as air or water, are Newtonian fluids.

While a fluid fundamentally comprises discrete molecules, at the scales this work is interested in fluids can be treated as a continuum, with a single fluid's properties varying smoothly in space. By using this continuum model the three variables defining the state of a fluid: its pressure, its velocity and its density, can be linked, and their variation over time defined using the Navier-Stokes equation (1.1):

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\nabla\cdot\mathbf{T}+\mathbf{f} \tag{1.1}$$

This equation makes no assumptions regarding the type of fluid it describes, and is derived from the basic principles of conservation of mass, momentum and energy.

In this work the evolution in time of fluids are to be modeled computationally using this equation. While other methods are available for CFD, all the methods within this work use a finite element method to discretize the domain over which we examine the fluid dynamics. This method splits the volume of the domain into many smaller sub-volumes, or cells, within which representative values of the fundamental fluid properties are calculated. By using values for a cell and its near-neighbours, the approximate differentials in space can be calculated. These differentials can then be used to advance the solution in time. The more cells there are, the closer this finite element method models a continuum, and hence the more accurate it becomes. However adding more cells also increases the computational resources required to advance them in time. For this reason it is desirable to develop methods that not only model the fluid as accurately as possible, based on the limited information available, but also allow a computer to make the most efficient use out of its limited resources.

Fluid flows can be categorized into two types: steady and unsteady. Steady flows are time invariant, and in CFD will tend to converge to a more accurate solution with simulation time. Conversely unsteady flows have a flow field that can vary significantly with time, and a longer simulation time will show these variations. Steady flows include flow around an aircraft wing at cruise, or slow speed flow through a pipe with a constant flow rate. Unsteady flows are more common in nature, for example a wave breaking over a rock or flow around the flapping wings of a bird. It should be noted that a flow can be both unsteady and stable, a case which is often characterized by oscillations around a fixed point. In this work the focus is almost entirely on unsteady stable flows.

1.2. Interfacial Flows

Of fluid dynamics phenomena visible on a day-to-day basis, the flow around interfaces is perhaps the most commonly witnessed. These flows range from very small flows, such as rain falling from the sky, to much larger scale flows such as waves on the ocean. The flows can be either stable, for example individual small bubbles rising in a carbonated drink, or unstable, such as a stream of water falling from a tap. They key property of all of these flows is that there is a clearly defined interface between the two phases and in a very small distance the properties of the flow can change hugely.

While all the examples given above are of air-water interfaces other interfaces are possible. So long as the two phases are adverse to mixing, flows with similar properties will be observed.

In this work the primary focus will be on air-water flows with length scales of the order 0.1mm to 10mm. In these flows a physical phenomena known as surface tension plays a very important role. Surface tension is a force, only apparent in some fluids, which becomes stronger the more curved an interface becomes. It is brought about by the interactions of molecules at the interface. In water, for example, each molecule is at its lowest energy state when in contact with many other water molecules. A molecule at the interface will hence have a higher energy state than molecules within bulk of the fluid. For the fluid as a whole to minimize its energy state the sum of the energies of the molecules at the interface must be

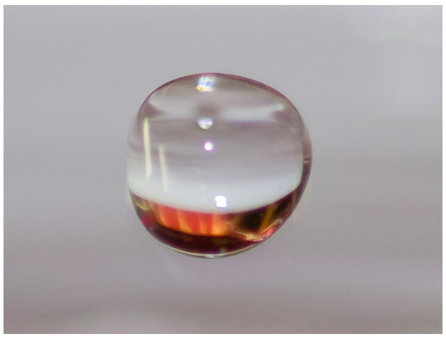

Figure 1.1.: The shape of rain droplets is determined by how the air flows around them as they fall. (Image: Omar Bariffi)

Figure 1.2.: Waves on the ocean are generated over large distances by the flow of air over the surface.

minimized. This leads to a minimization of molecules at the interface, and hence a minimization of interface length, and a minimization of curvature [2].

At the length scales of interest to this work surface tension can be modeled by a pressure jump at the interface. The size of this jump (Δp) can be calculated using the Young-Laplace Equation (1.2) [3, 4]:

$$\triangle p = \sigma(\kappa_1 + \kappa_2) \tag{1.2}$$

Where σ is the surface tension coefficient of the interface defined in $N.m^{-1}$, and κ_1 and κ_2 are orthogonal curvatures with radii of curvature normal to the interface. The surface tension coefficient is a property of the interface and varies depending on the composition and state of the fluids either side.

Besides the pressure jump caused by surface tension incompressible flows have two other significant properties discontinuous at the interface: density and viscosity. For an air-water interface the density ratio is approximately 1:800, while the viscosity ratio is approximately 1:55. These discontinuities can be particularly hard to model using CFD as many traditional approaches require the fluid equations to be continuous. A significant part of this work will be dedicated to the modeling of the interface, and in particular the modeling of these discontinuities.

1.3. Graphics Technology and its Application to Computational Fluid Dynamics

Developments in Graphics Processing Units (GPUs) have recently allowed for them to be easily used as general purpose massively parallel (thousands of concurrently running operations) computing devices. While these advances were originally intended to allow complex visual effects to be displayed using a large number of pixels for computer graphics, it was found that the same technology could be applied to perform calculations for scientific computing, often much more efficiently than traditionally used Central Processing Units (CPUs).

One of the first scientific applications for GPUs was presented by Lengyel et al. in 1990 and concerned robot motion planning [5]. Many other applications have since followed [6, 7, 8, 9]. At first the tools were extremely limited and programmers had to treat each parallel operation as analogous to a pixel on the screen, or a vertex on a three dimensional object. It was not until the release of NVIDIA's Compute Unified Device Architecture (CUDA) programming language in 2007 that scientific computing on GPUs finally became fully decoupled from the "graphics" model [10].

Although many applications make use of GPUs, the field is still relatively young, with the first double precision capable GPU released only a few years ago in 2008. Despite its youth GPUs do seem to be gaining traction in the HPC community. Table 1.1 shows the growth of the CUDA programming language from 2008 to May 2012.

Year:	2008	2012
CUDA SDK Downloads	150,000	1,500,000
Supercomputers	1	35
Universities teaching CUDA	60	560
CUDA Academic Papers	4000	22,500

Table 1.1.: CUDA by the numbers [11].

There are two primary reasons for the decision to develop the code used for this work on GPUs:

- The development of methods which run well on GPUs is itself an interesting scientific subject. It is not always possible to directly convert methods that are efficient on CPUs to GPUs and the challenges that arise from this are worthy of study in themselves.
- GPUs, by their nature, can often run codes more efficiently than CPUs. Cranfield's small cluster comprising eight GPUs has been shown to perform on-par with over 200 individual CPU cores at a fraction of the power requirements, when methods that map well to the GPU are used. The quality of the results from CFD simulations is often limited by the amount of processing power that can be applied to a given problem and increasing this power is obviously desirable.

While GPUs can be used to solve many problems, it is important to remember that they have their limitations. Often a model simply cannot be made to run efficiently in a massively parallel environment. Furthermore, converting a CPU program to run on GPUs can take a significant amount of time which may not be worthwhile in some cases.

1.4. Droplet Dynamics

It is intended that the models developed for simulation of interfacial flows will be used to model the behaviour of water droplets falling freely though air. These water droplets have properties akin to naturally occurring rainfall at sea level, and this was, in-part, the inspiration for this study. The focus of the simulations will be on droplets of $1mm$ to $3mm$ spherical radius.

It is a common misconception that rain drops form a "teardrop" type shape as they fall. Instead, the shape greatly depends on the size of the droplet. Small droplets tend to both fall more slowly (due to having a higher ratio of viscous to inertial forces) and have higher mean curvature, and therefore a higher a surface tension. The combination of a lower external pressure and a higher restoring force means that droplets of radius up to $1mm$ tend to be roughly spherical. Larger droplets, however, deform much more significantly. This deformation tends to result in a flattening of the droplet as a whole, particularly at the base. This change in deformation with size is shown diagrammatically in Figure 1.3.

While these deformations give an idea as to the average shape of a droplet as it falls, they do not tell the whole story. Droplets of greater than 0.5mm radius have been found to oscillate in multiple low frequency modes with few significant harmonics at higher frequencies [13]. There are also other properties of interest, such as the terminal velocity, the circulation of the water inside the droplet, and the shape, size and behaviour of that wake.

The set of simulations to be run have never previously been attempted computationally and it is hoped that they will be able to bring new insight into droplet dynamics. The requirements of the numerical model used to simulate the processes surrounding droplets at terminal velocity are surprisingly taxing. The combination of significant higher order spatial derivatives of the velocity field in the near interface region, and very large discontinuities in density and viscosity at the interface, combine to form relatively subtle movements in that interface. Even slight inaccuracies in the model can lead to significant errors. As such, a powerful numerical model is required to produce accurate results.

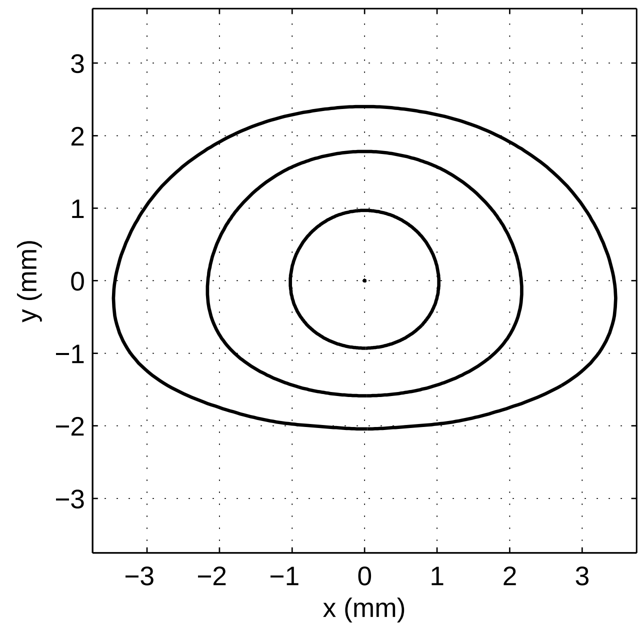

Figure 1.3.: Deformation of droplets with equivalent radius 1mm (inner), 2mm (centre) and 3mm (outer). These two dimensional profiles are based on the equilibrium shape profiles theorized by Beard and Chuang [12]. The equilibrium profile of the 1mm radius droplet has an axis ratio (horizontal:vertical chord length at the broadest point) of approximately 1.09:1, the 2mm droplet 1.29:1 and the the 3mm droplet 1.56:1.

1.5. Thesis Objectives

There are three main objectives for this work, each of which has a chapter primarily concerned with it:

1. To present a 3D flow solver able to simulate the interaction of two immiscible fluids at low speeds. This solver is designed to simulate unsteady incompressible gravity and curvature driven flows, as accurately as possible (Chapter 2).

2. To demonstrate the performance that can be attained by using GPUs to execute CFD problems of this type and compare this to conventional CPU performance (Chapter 3).

3. To demonstrate this GPU accelerated 3D flow solver by simulating water droplets falling under gravity and, in doing so, bring new insight into the behaviour of these droplets (Chapter 4).

Chapter 5 concludes the thesis and makes suggestions for potential future research.

1.6. Publications and Related Work

So far a single publication has been made as a direct result of this work:

- **J. Appleyard** and D. Drikakis. 'Higher-order CFD and interface tracking methods on highly-Parallel MPI and GPU systems'. Computers & Fluids, Volume 46, Issue 1, July 2011, Pages 101–105

The following paper is currently being written and submission is planned imminently.

- **J. Appleyard** and D. Drikakis. 'CFD Investigation into the properties of water droplets falling through air'

Related work not directly resulting from this academic study, but completed in parallel by the author:

- J.R. Appleyard, **J.D. Appleyard**, M. Wakefield, A. Desitter. 'Accelerating Reservoir Simulators using GPU Technology'. SPE Reservoir Simulation Symposium, 21-23 February 2011, The Woodlands, Texas, USA.
- Presentation on 'New Ideas for Massively Parallel Preconditioners (S0432)' at NVIDIA GTC 2012, San Jose, California.

This related work was completed in the author's role as a consultant analyst for Polyhedron Software Ltd.

2. Numerical Methods

Synopsis

Numerical simulations of fluids require specialized methods to model them accurately and quickly. This project aims to model low speed immiscible fluids with a large density discontinuity at their interface. This chapter details the methods used to model these flows. Part of the code used for this project was based on Hirecom, a FORTRAN 90/95 code developed by Cranfield's Department of Engineering Physics. Hirecom contains implementations of the simpler parts of the methods presented here. More specifically, at the initiation of the project Hirecom was capable of solving low density ratio incompressible miscible flows. It was also capable of solving zero surface tension immiscible flows compressibly. The changes to Hirecom for the current work required reworking implementations throughout the code, as many were found to be implemented correctly only in a narrow operating domain. In the first part of this chapter the governing equations for low Mach flows and methods for interface tracking are presented. The second part describes the pressure projection technique used to enforce incompressibility within the flow and the methods used to couple this with the interface tracking method, to ensure the flows remain unmixed.

2.1. Governing Equations

The aim of these methods is to simulate low Mach interaction between two or more immiscible fluids with a density ratio of at least three orders of magnitude. It is assumed that density and viscosity are uniform within each fluid, and that each is Newtonian. The equations of motion for this system in three dimensions are:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \frac{(\nabla \cdot \tau)^T}{\rho} \tag{2.1}$$

where:

$$\begin{aligned} \mathbf{u} &= (u, v, w)^T \\ \mathbf{g} &= (0, 0, g)^T \end{aligned} \tag{2.2}$$

$$\tau = \mu \left(\begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix} + \begin{pmatrix} \nabla u \\ \nabla v \\ \nabla w \end{pmatrix}^T \right) \tag{2.3}$$

To obtain the solution at the next time step Equation 2.1 is advanced forward in time in two stages. Firstly the solution is advanced neglecting the pressure term:

$$(\rho \tilde{\mathbf{u}})^{n+1} = (\rho \mathbf{u})^n - \triangle t \left(\rho \left((\mathbf{u} \cdot \nabla)\, \mathbf{u} + \mathbf{g} \right) + (\nabla \cdot \tau)^T \right) \tag{2.4}$$

The pressure term is then calculated implicitly so as to conserve mass at the $n+1$ time step:

$$(\rho \mathbf{u})^{n+1} = (\rho \tilde{\mathbf{u}})^{n+1} - \triangle t \cdot \nabla p \tag{2.5}$$

The steps for solving first Equation 2.4 and then Equation 2.5 numerically will be described in this chapter. Particular attention will be placed on the treatment of the jump conditions arising at the interface between the immiscible fluids, as many of the differentials above are not smooth in the near-interface region.

2.2. Spatial Integration

To solve the equations given in Section 2.1 the domain is modeled as a set of discrete three dimensional cells. Each of these cells has several floating point values associated with it, which represent the mean values of the fluid properties over the volume contained within the cell. In this case these properties are the pressure, density, distance from the interface and the x, y and z momenta per unit volume. As the problem has no fixed geometry to map the grid to, the grid is structured with cuboidal cells whose faces align with the three Cartesian axes. In this section the mapping of Equation 2.4 to this discrete grid will be described.

2.2.1. Interface Tracking

As the fluids are immiscible there is a clearly defined interface between them. This interface is able to move and therefore cannot be assumed to be aligned with a static grid. For this reason the interface position must be tracked so that properties of the field any point can be determined. Furthermore it is advantageous if properties such as normals and curvature can be easily determined from the interface tracking method, as there are physical phenomena (specifically, in this case, surface tension) that rely on them.

For the above reasons, it has been decided to use the level set method to track the interface. The level set method was first described by Osher and Sethian [14] and has been used by many authors since [15, 16, 17]. For this method the scalar function $\phi(x,t)$ is introduced. This function is initialized as a signed distance function, the magnitude of which is set to be the distance from the interface, and the sign to be positive for regions outside of the fluid, and negative for regions inside. The position of the interface is therefore defined by the isosurface given by $\phi(x,t) = 0$. This function is advanced in space and time by solving the level set equation using level set velocity $\mathbf{u}_{LS}$:

$$\frac{\partial \phi}{\partial t} + \mathbf{u}_{LS} \cdot \nabla \phi = 0 \tag{2.6}$$

In this work the level set equation is solved using a High-Order Upstream Central (HOUC) finite element scheme [18] (simple upwinding expanded to higher orders) to update ϕ at each time step. These schemes have been found to be more efficient than the more usual Essentially Non-Oscillatory (ENO) or Weighted Essentially Non-Oscillatory (WENO) schemes and, in the case of a smooth level set function, cause no detriment to numerical stability. In this work, unless otherwise stated, the 5th order HOUC scheme described in the x-direction by Equation 2.7 is used for the spatial dicretization of the level set equation.

$$\phi_x \approx \begin{cases} \frac{1}{60}\left[-2\phi_{i-3} + 15\phi_{i-2} - 60\phi_{i-1} + 20\phi_i + 30\phi_{1+1} - 3\phi_{i+2}\right] & u_{LS} > 0 \\ \frac{1}{60}\left[2\phi_{i+3} - 15\phi_{i+2} + 60\phi_{i+1} - 20\phi_i - 30\phi_{1-1} + 3\phi_{i-2}\right] & u_{LS} < 0 \\ 0 & u_{LS} = 0 \end{cases} \quad (2.7)$$

As previously stated, one of the key advantages of the level set method is that it allows for easy calculation of the properties of the interface. The normal field $\mathbf{N}(x, y, z)$ can be calculated, at any point, directly from the level set equation using Equation 2.8:

$$\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|} \quad (2.8)$$

The curvature field $\kappa(x, y, z)$ is the calculated by taking the divergence of the normal:

$$\kappa = -\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) \quad (2.9)$$

Expanding Equation 2.9 in two Cartesian dimensions gives the following equation for curvature:

$$\kappa = -\frac{(\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx})}{(\phi_x^2 + \phi_y^2)^{1.5}} \quad (2.10)$$

In this work the differentials of the level set field for normal and curvature calculation are computed using central differences.

As several authors have noted discretizing this equation using central differences can lead to spurious results for curvature if the level set field is not smooth [19, 20]. One solution to this problem is to reconstruct the interface on to a local sub-grid [19]. Another is to use a least squares method to attempt to smooth any irregularities in the level set field [20].

In this work a hybrid of Equation 2.10 and a slightly modified version of the least squares method presented by Marchandise et al. is used [20]. It was found that direct implementation of this method resulted in some significant errors, especially in regions where the curvature field changes sign multiple times in a small area. These changes of sign can be incorrectly smoothed into a significantly lower curvature, resulting in potential instabilities of high frequency modes.

For these reasons several changes have been made to the implementation of Marchandise et al. with the intention of retaining its smoothing properties where required, but suppressing them where not. A quality parameter q is introduced to quantify the smoothness of the local level set field:

$$q = |1 - \sqrt{\phi_x^2 + \phi_y^2 + \phi_z^2}| \tag{2.11}$$

With κ_{sq} representing the least squares curvature and κ_{lvl} the level set curvature, the final calculated curvature is given by Equation 2.12:

$$\kappa = \begin{cases} \kappa_{lvl} & q \leq 0.05 \\ \frac{\theta\kappa_{sq}+(2-\theta)\kappa_{lvl}}{2} & 0.05 < q < 0.5 \\ \frac{\kappa_{sq}+\kappa_{lvl}}{2} & q \geq 0.5 \end{cases} \tag{2.12}$$

where:

$$\theta = \frac{(q-0.05)}{0.45} \tag{2.13}$$

This interpolation method goes some way toward smoothing the irregularities in the level set field, without leading to instability.

Conservative level set

One of the drawbacks to the level set method has traditionally been its poor conservation properties in the near-interface region [21]. The reasons for this are twofold. Firstly, mass in an incompressible fluid is proportional to the volume of that fluid. The level set method is not designed to conserve volume, so errors are likely to accumulate over time. Secondly, the values of the conserved properties (ie. momentum and density) are discontinuous at the interface. Differentiating over these discontinuities clearly leads to incorrect results. One proposed solution to this problem is to combine Ghost Fluid Method (GFM) [15] with the level set method to help reduce these conservation errors. In this method *ghost* values of the fluid properties of each phase are calculated in cells which lie across a boundary from the working cell, and these *ghost* values are then used in place of the real values when differentiating.

While the GFM helps to reduce the errors due to differentiating across an interface it does not eliminate them. An alternative conservative method first described by Hu et al. in two dimensions [22], and more recently extended into three dimensions by Barton et al. [23], is used in this work. This method achieves conservation by reconstructing the position of the interface within mixed cells, and then solving the inviscid momentum equations for each phase within each cell individually, as if the cell were cut in two. The solution at the interface is then calculated by solving a Riemann problem on the interface plane.

A drawback of this type of conservative level set method is that it has the potential to create very small sub-cells, which can restrict the maximum time step to correspondingly small values. To overcome this problem a cell merging method is used. Any sub-cell whose volume is less than a certain threshold is merged with a neighbouring sub-cell whose volume is greater than that same threshold volume. In this work the threshold volume is chosen to be half of the volume of the smallest cell in the domain. While this does result in some loss of resolution in the near interface region the improved conservation properties make this trade-off worthwhile.

A brief description of the method used to reconstruct the interface and to merge cells is presented below. For further details the reader is directed to [22] and [23].

Interface Reconstruction

To calculate accurate volume fractions and cell face apertures the interface must be reconstructed from the level set field. This reconstruction requires the following steps to be performed on each mixed cell:

1. Compute the values of the level set field at each of the cell's corners. These values are computed by taking the mean of the values of the level set field in neighbouring eight cells.
2. Using linear interpolation of these averaged values, find the points where the interface intersects the cell edges (ie. where $\phi = 0$).
3. The fraction of a given face which lies within the phase can be determined by calculating the area A_i of the triangle, or quadrilateral, with corners at the interface intersections calculated in step 2.
4. Having calculated these fractions the area of the interface plane within the cell can be calculated using the Pythagorean theorem.
5. The volume contained within the interface can then be calculated by using the sum of tetrahedra method for volumes of arbitrary polyhedra. Where

the vector $\mathbf{x}_i$ is a vector from an arbitrary point to the centroid of the face, and $\mathbf{n}_i$ the unit normal facing outward from the enclosed volume, any polyhedron's volume can be determined using Equation 2.14:

$$V = \frac{1}{3}\left(\sum_{i=1}^{N_{faces}} (\mathbf{x}_i \cdot \mathbf{n}_i)\, A_i\right) \tag{2.14}$$

It should be noted that this reconstruction method assumes that the interface is planar. In regions of extremely high curvature this assumption will lead to errors in the volume fraction and apertures calculated, and should be considered when analyzing results where radii of curvature approach the resolution of the grid.

It is also assumed that the mixed cell only contains one interface plane. In cases of merging or splitting phases this is often an incorrect assumption. As this work does not aim to simulate such flows, this is unlikely to cause errors, however should an extension be required at a later date to more accurately model such flows the Marching Cubes method of Lorensen and Cline could be used to differentiate between the interface planes [24].

Cell Merging Method

As previously stated, to prevent the time step from becoming prohibitively small, any sub-cell with a volume fraction, α, of less than half of the volume of the smallest cell in the domain is merged with a neighbouring sub-cell. These low volume fraction sub-cells are termed *small cells.* At the start of each time step the following procedure is carried out:

1. For all small cells choose a neighbouring *target cell.* This cell should be a neighbour of the small cell, and should ideally lie in the normal direction. Each target cell may be shared by any number of small cells.
2. Merge the small cells with their target cells. This is achieved by updating the target cell's momenta to be a volume fraction weighted average of the values of momenta defined at the target cell, and those of the small cells being merged into it. To update the target cells (subscript T) is with their associated small cells (subscript S) Equation 2.15 is used.

$$(\alpha\mathbf{u})_T^* = (\alpha\mathbf{u})_T + \frac{\alpha_T \sum^{N_S} (\alpha\mathbf{u})_S - (\alpha\mathbf{u})_T \sum^{N_S} \alpha_S}{\alpha_T + \sum^{N_S} \alpha_S} \tag{2.15}$$

3. Set the small cells' momenta and density equal to that of their target cell's. The cells are now merged.

The overall mass and momentum of the merged cells remains unchanged by this procedure and, as the cells are effectively one, the CFL condition now applies to the merged cell rather than the small cell, removing the time step constraint the small cell was previously causing. It should be noted that when using a multiple stage time integration method (such as the Runge-Kutta method used here and described later) it is important to maintain the cell pairings through all stages of a single time step.

Interface Velocity

The level set velocity for use in Equation 2.6 can be calculated in a number of ways. The simplest method is to equate the level set velocity at a point with the velocity of the fluid at that point. This method is acceptable, but tends to distort the signed distance function, especially in regions where the velocity gradient normal to the interface is high. An alternative method is to calculate the level set velocity only at the interface and then extrapolate it into the far field. This greatly reduces the level set velocity gradient in the normal direction resulting in much less compression or rarefaction of the signed distance function.

In this work, the velocity at the interface $\mathbf{u}_I$ is defined in all the cells where both phases are present as the density weighted average of phase velocities:

$$\mathbf{u}_I = \frac{\rho_1 \mathbf{u}_1 + \rho_2 \mathbf{u}_2}{\rho_1 + \rho_2} \tag{2.16}$$

This velocity is then extrapolated to all the cells in the domain using the method described by [23]: a quantity q can be extrapolated along unit level set normals $\mathbf{n}$ by solving Equation 2.17 to steady state in pseudo-time τ:

$$\frac{\partial q}{\partial \tau} \pm \mathbf{n} \cdot \nabla q = 0 \tag{2.17}$$

Reinitialization

Despite the level set velocity extrapolation method described above, the level set field will still deviate slowly from a signed distance function. To solve this, the level set field must periodically be reinitialized. Here, the High-order Constrained Reinitialization (HCR-2) method presented by Hartmann et al. is used [25].

As Walker and Müller observe this constrained reinitialization method will tend to reduce the volume contained within a convex shape [26]. As the model is incompressible, this reduction in volume results in an unphysical reduction in mass. For this reason, a small correction must be made on each reinitialization to offset this error: if the mass of a phase drops below 99% of it's initial mass after reinitialization, the following fixed offset to the level set field is applied:

$$\phi = \phi_0 + 0.02 \cdot \triangle x \tag{2.18}$$

While more complicated solutions to this problem could be found, this correction is very low in computational expense, and was found, in all tested cases, to adequately offset the mass loss due to reinitialization without over-correcting.

2.2.2. Inviscid Terms

By defining A as the area on which the flux is applied and V as the volume of the relevant cell, the momentum changes due to the inviscid and gravity terms of Equation 2.4 can be discretized as follows:

$$(\Delta(\rho\mathbf{u}))_{inv} = -\Delta t\rho\left((\mathbf{u}\cdot\nabla)\mathbf{u} + \mathbf{g}\right) \approx -\Delta t \cdot \rho\left(\frac{\sum_{i=1}^{N_{Faces}}\left(A_i\left(\mathbf{n}_i \cdot \mathbf{u}_i\right)\mathbf{u}\right)}{V} + \mathbf{g}\right) \tag{2.19}$$

where the unit normals, $\mathbf{n}_i$, point outward from the cell faces. The implementation of the gravity terms is trivial as it simply requires a constant velocity change to all cells in the domain. Discounting gravity, and considering only fluxes on faces aligned in the x-direction, Equation 2.19 can be expanded by employing Godunov's conservative finite volume method [27] to:

$$(\Delta(\rho\mathbf{u}))_{inv} \approx -\Delta t \cdot \rho\left(\frac{\left((A\mathbf{E})_{i+\frac{1}{2}} - (A\mathbf{E})_{i-\frac{1}{2}}\right)}{V}\right) \tag{2.20}$$

where:

$$\mathbf{E} = (u^2, uv, uw)^T \tag{2.21}$$

The flux, $\mathbf{E}$, is computed by solving the Riemann problem at the cell face. To solve the Riemann problem, the states of the variables on the left and right sides of the face must first be defined. In this work these variables are reconstructed from the surrounding flow field using a 5th order Kim-Kim limiter [28] as $\mathbf{U}_L$ and $\mathbf{U}_R$. A Rusanov Riemann solver [29] is then used to solve for the fluxes at the cell interfaces. The Rusanov Riemann solver defines these fluxes in the x-dimension as:

$$\mathbf{E}^{RU}_{i+\frac{1}{2}} = \frac{1}{2}\left((\mathbf{E}_L + \mathbf{E}_R) - s\left(\mathbf{U}_R - \mathbf{U}_L\right)\right) \tag{2.22}$$

where:

$$s = max\left(|u_L|, |u_R|\right) \tag{2.23}$$

To solve for faces not aligned with the x-direction, a change of coordinate systems is required. In the case of faces aligned along either of the other Cartesian axes this procedure is trivial. For example fluxes, $\mathbf{F} = (uv, v^2, vw)^T$, aligned with the y-axis are calculated using:

$$\mathbf{F}^{RU}_{j+\frac{1}{2}} = \frac{1}{2}\left((\mathbf{F}_L + \mathbf{F}_R) - s\left(\mathbf{U}_R - \mathbf{U}_L\right)\right) \tag{2.24}$$

where:

$$s = max\left(|v_L|, |v_R|\right) \tag{2.25}$$

Some additional care is required in the near interface region when implementing this method. Firstly, the phase interface is treated as a cell boundary, but is not aligned with the grid. This means that the Kim-Kim limiter cannot be applied at the phase interface. It is instead treated in a first-order manner, with left and right states simply defined as the values of the properties within the cells to the left and the right of the interface in the rotated co-ordinate system. Secondly, while the Kim-Kim limiter is designed to treat flow fields with significant high order terms (such as a change in phase) smoothly, calculating differentials across the interface in unnecessary. Instead, in any cell where the flow properties of a particular phase are not defined, the input into the Kim-Kim reconstruction method is set equal to the value closer to the interface from it. For example, if the cell at $x+2$ lies fully outside of a phase, the value for this cell at $x+1$ is used.

This method is an implementation of a method known as Implicit Large Eddy Simulation (ILES). In explicit Large Eddy Simulation (LES) an explicit subgrid model is required to model the decay of turbulence at scales smaller than the grid spacing, whereas in ILES the numerics are designed such that modeling of these sub-grid scales are implicit. For further details the reader is directed to [30].

2.2.3. Viscous Terms

As the Reynolds numbers of the flows studied in this work are expected to be relatively high ($Re > 500$), the viscous momentum fluxes are expected to be small in comparison to the convective terms. Although sharp viscosity implementations exist [31], none are compatible with the conservative level set method. Sharp treatment of viscosity in the near interface is desirable however it was decided that, as interface viscosity is not expected to be a major contributor to the overall flow field, rather than developing a new method to handle it sharply, it would be treated as a smoothly varying property at the interface.

The term to be discretized is given in Equation 2.26:

$$\left(\Delta\left(\rho\mathbf{u}\right)\right)_{vis} = -\Delta t \cdot \left(\nabla\cdot\tau\right)^{T} \tag{2.26}$$

With τ being the incompressible viscous stress tensor as defined in Equation 2.3. As viscosity is defined to be smooth across the interface a continuous viscosity is defined using a Heaviside step function:

$$\mu_c\left(\phi\right) = \mu_2\left(\left(\frac{\mu_1}{\mu_2}\right)^{H(\phi)}\right) \tag{2.27}$$

where:

$$H\left(\phi\right) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1+sin\left(\frac{\pi\phi}{2\epsilon}\right)}{2} & -\epsilon \leq \phi \leq \epsilon \\ 1 & \phi > \epsilon \end{cases} \tag{2.28}$$

This interpolation method has two key properties:

- It has a continuous gradient which reduces the potential for oscillations resulting from small changes.
- At the interface (ie. $\phi = 0$) the viscosity is given by $\mu_I = \sqrt{\mu_1\mu_2}$ resulting in the equality: $\frac{\mu_I}{\mu_1} = \frac{\mu_2}{\mu_I}$. This property means that the momenta of the two phases are adversely affected to the same degree by errors in viscosity.

No reference could be found using this interpolation method and it is thought to be novel.

For the Heaviside step function ϵ should be chosen to be of the same order as the grid spacing, though no lower. In this work ϵ is set equal to 1.5 times the smallest grid spacing in the domain.

The divergence operator in Equation 2.26 is discretized using a finite volume method. For clarity only the fluxes resulting from expanding the divergence operator in the x-direction is presented:

$$\left(\Delta\left(\rho\mathbf{u}\right)\right)_{x,vis} \approx -\frac{\left(\mu_c\begin{pmatrix}2u_x\\ u_y+v_x\\ u_z+w_x\end{pmatrix}\right)_{i+\frac{1}{2}} - \left(\mu_c\begin{pmatrix}2u_x\\ u_y+v_x\\ u_z+w_x\end{pmatrix}\right)_{i-\frac{1}{2}}}{\Delta x} \tag{2.29}$$

The differentials are discretized directly using central differencing. For example:

$$\left(u_x\right)_{i+\frac{1}{2}} \approx \frac{u_{i+1}-u_i}{\Delta x} \tag{2.30}$$

$$\left(u_y\right)_{i+\frac{1}{2}} \approx \frac{u_{i+1,j+1}+u_{i,j+1}-u_{i+1,j-1}-u_{i,j-1}}{4\Delta y} \tag{2.31}$$

2.2.4. Reference Frame

As the movement of an interface under gravity is to be studied, the choice of reference frame is important. A stationary reference frame, for example, would require the domain to span the entire volume through which the interface would move. Given that sub millimeter resolutions, traveling over distances of several meters are to be modeled, it is clear that a stationary reference frame would lead to very high computational requirements.

For this reason a reference frame such that the mass contained within the interface is constrained to move only slowly relative to the reference frame is used. While it is possible to constrain the centre of mass of the interface almost exactly, this is undesirable as it can lead to accumulation of discretization errors over time. For this reason a slightly more relaxed constraint is used, defined in the x-dimension by Equation 2.32 and extended in the natural manner to the y and z dimensions:

$$u_{correct,x} = \begin{cases} \frac{\alpha(\Delta t)\cdot(x_{COM}-\beta)}{\beta} & |x_{COM}| > \beta \\ 0 & otherwise \end{cases} \tag{2.32}$$

where α is a scaling factor, β is a threshold distance and x_{COM} is the position of the centre of mass when projected on to the x-axis. While the values of these parameters are clearly dependent on the problem, it was found through experimentation that values of approximately $\alpha = 0.25$ and $\beta = \Delta x$ maintain the interface in a central location in the falling droplet test case presented in Chapter 4 without correcting too aggressively. This correction is calculated in all three dimensions and the velocity $\mathbf{u}_{correct}$ added to every cell in the domain, accumulating as a constant normal velocity boundary condition at the domain edges.

2.3. Temporal Integration

The time derivatives in Equation 2.1 are integrated using the third-order Total Variation Diminishing (TVD) Runge-Kutta method described by Shu and Osher [32].

The overall time step used for each iteration must be based upon all the forces that could act on the fluid in a cell. Convection, viscosity, gravity and surface tension must all be considered. The analysis of Kang et al. suggests an appropriate time step restriction can be calculated from the following equations [31]:

$$C_t = \frac{1}{\alpha}\left(\frac{|u|_{max}}{\Delta x} + \frac{|v|_{max}}{\Delta y} + \frac{|w|_{max}}{\Delta z}\right) \tag{2.33}$$

$$V_t = \frac{max\left(\frac{\mu_1}{\rho_1}, \frac{\mu_2}{\rho_2}\right)}{\alpha^2}\left(\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} + \frac{2}{(\Delta z)^2}\right) \tag{2.34}$$

$$G_t = \sqrt{\frac{|g|}{\alpha \Delta z}} \tag{2.35}$$

$$S_t = \sqrt{\frac{\sigma\,|\kappa_{max}|}{\alpha^2 \cdot min\,(\rho_1, \rho_2) \cdot (min\,(\Delta x, \Delta y, \Delta z))^2}} \tag{2.36}$$

With a final time step of:

$$\Delta t = \frac{2 \cdot CFL}{(C_t + V_t) + \sqrt{(C_t + V_t)^2 + 4\left((G_t)^2 + (S_t)^2\right)}} \tag{2.37}$$

Where α is the minimum unmixed volume fraction within a cell. As any cells with volume fractions of less than a half are mixed in this work, $\alpha = 0.5$.

In all the simulations reported in the results section the CFL number was 0.5.

2.4. Pressure Projection Method

The pressure projection method was introduced by Chorin [33, 34]. The first work adapting this method to incompressible flows published by Bell et al., who developed an unsteady second order constant density solver [35]. This work was later extended to incompressible variable density flows [36]. Further extensions were made by Shu et al., who implemented spectral methods for the solution of the pressure field [37], and by Andrews who approached the problem using a fractional-step method [38]. While the pressure projection method has been previously coupled with the level set method for immiscible incompressible flows [31], the combination of the conservative level set method described in Section 2.2.1 and the pressure projection method requires some additional consideration at points.

The aim of the pressure projection method is to decompose the velocity field given by the solution of Equation 2.4 ($\tilde{\mathbf{u}}^{n+1}$) into the sum of its divergence free part $\mathbf{u}^{n+1}$ (ie. the solution after the time step) and the gradient of a scalar field ϕ.

$$\tilde{\mathbf{u}} = \mathbf{u} + \sigma \nabla \phi \tag{2.38}$$

By substituting $\phi = p$ and $\sigma = \frac{\Delta t}{\rho}$ Equation 2.38 can be modified to fit the form of Equation 2.5. By taking the divergence of both sides of this equation and applying the incompressibility constraint: $\nabla \cdot \mathbf{u} = 0$, Equation 2.39 is reached:

$$\nabla \cdot \left(\frac{\nabla p}{\rho}\right) = \frac{\nabla \cdot (\tilde{\mathbf{u}})}{\Delta t} \tag{2.39}$$

The solution to this equation yields a pressure field which can be projected using Equation 2.5 to give the incompressible solution.

Depending on the discrete form of the left hand side of Equation 2.39 the pressure projection method can be either exact or approximate. The exact formulation, while more computationally expensive, strictly enforces the divergence free criterion. The approximate formulation tends to be more difficult to implement as it requires additional filters in order to reduce the errors brought about by the approximations [30]. The exact formulation has been chosen for this work.

Each side of Equation 2.39 must be discretized separately. The left hand side is discretized in two dimensional Cartesian co-ordinates as follows using standard central differences:

$$\begin{aligned}\nabla \cdot \left(\frac{\nabla p}{\rho}\right) &= \nabla \cdot \left(\frac{1}{\rho_{i,j}}\begin{pmatrix}\frac{p_{i+1/2,j}-p_{i-1/2,j}}{\triangle x}\\ \\ \frac{p_{i,j+1/2}-p_{i,j-1/2}}{\triangle y}\end{pmatrix}\right)\\ &= \frac{1}{(\Delta x)^2}\left(\frac{p_{i+1,j}-p_{i,j}}{\rho_{i+1/2,j}}-\frac{p_{i,j}-p_{i-1,j}}{\rho_{i-1/2,j}}\right)+\\ &\quad \frac{1}{(\Delta y)^2}\left(\frac{p_{i,j+1}-p_{i,j}}{\rho_{i,j+1/2}}-\frac{p_{i,j}-p_{i,j-1}}{\rho_{i,j-1/2}}\right)\end{aligned} \tag{2.40}$$

where the inter-cell densities $\rho_{i\pm 1/2,j}$ and $\rho_{i,j\pm 1/2}$ are defined using the method outlined in section Section 2.4.1.

When discretizing the right hand side care must be taken as the velocity gradient across the interface can be discontinuous. The definition of the discrete divergence of the velocity in a given volume V is:

$$\nabla \cdot (\tilde{\mathbf{u}}) = \frac{\sum_{i=1}^{N_{faces}} A_i\left(\mathbf{u}_i \cdot \mathbf{n}_i\right)}{V} \tag{2.41}$$

In the far field, the solution to this equation is equivalent to taking central differences and assuming that interface velocities are simply the mean of the cell velocities either side of the interface. However in split cells this assumption is invalid and instead the divergence must be calculated from the full equation.

The value for the pressure has been chosen to be defined only once at each node, and as such the pressure projection method can only work on a single phase per node. For this reason methodology from the cell merging method used to maintain stability of the inviscid fluxes must be borrowed. As the phase at the cell centre is the only one that can be corrected, any partial cells which are not to be corrected must be merged with a target which is to be corrected, otherwise mass and momentum are not conserved. The same cell merging method described in Section 2.2.1 is applied to these cells. It is also important here that Equation 2.41 is computed by taking all the faces of the fully merged cell.

Once both the left and right hand side of Equation 2.39 have been calculated, a linear series of equations remains. These equations can be represented in matrix form by the equation:

$$\mathbf{A}\mathbf{p} = \mathbf{b} \tag{2.42}$$

where the solution to this equation for the vector $\mathbf{p}$ is sought. The methods required to solve such a system have long been a topic of discussion in the scientific community the details of which are beyond the scope of this work. For an overview the reader is directed to [39]. The Bi-Conjugate Gradient Stabilized method (BiCGStab) was chosen to calculate the solution in this project, the details of which will be described later in this chapter (Section 2.4.2). Once the pressure field has been found, the final step is to project it on to the intermediate velocity field to obtain a divergence free velocity field.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \triangle t \cdot \left(\frac{\nabla p}{\rho}\right) \tag{2.43}$$

The $\frac{\nabla p}{\rho}$ term in Equation 2.43 is discretized by taking the mean of the term at cell faces along the relevant axis:

$$\frac{\nabla p}{\rho} = \begin{pmatrix} \frac{1}{2\Delta x}\left(\frac{p_{i+1,j}-p_{i,j}}{\rho_{i+1/2,j}} + \frac{p_{i,j}-p_{i-1,j}}{\rho_{i-1/2,j}}\right) \\ \frac{1}{2\Delta y}\left(\frac{p_{i,j+1}-p_{i,j}}{\rho_{i,j+1/2}} + \frac{p_{i,j}-p_{i,j-1}}{\rho_{i,j-1/2}}\right) \end{pmatrix} \tag{2.44}$$

It is important to discretize it in this way rather than by simply using the cell centred density, as the $\frac{\nabla p}{\rho}$ term used here must be the same as that which was calculated in Equation 2.40.

The procedure is therefore as follows:

1. Apply the cell merging method for small cells. Additionally, any any cell with $\phi < 0$ must be merged with a valid neighbouring cell.
2. Calculate the velocity divergence across each cell and merged cell using Equation 2.41.

3. Calculate the pressure field required to produce a divergence free solution Equation 2.42.
4. Project these pressures on to the velocity field of all cells with $\phi \geq 0$ using Equation 2.43.

By applying this variation on the cell merging method presented earlier, this new procedure allows the conservative level set method and the pressure projection method to be used in tandem.

2.4.1. Treatment of Jump Conditions

At the interface there are two spatial jump conditions which much be considered when calculating Equation 2.40: a pressure jump and a density jump. These are defined at the interface by Equation 2.45 and Equation 2.46:

$$[p]_\Gamma = \sigma\kappa_I \tag{2.45}$$

$$[\rho]_\Gamma = \rho^+ - \rho^- \tag{2.46}$$

Considering the Poisson equation to be solved (Equation 2.39) simplified into the one dimensional case:

$$\frac{1}{(\Delta x)^2}\left(\frac{p_{i+1}-p_i}{\rho_{i+1/2}} + \frac{p_i - p_{i-1}}{\rho_{i-1/2}}\right) = RHS \tag{2.47}$$

If an interface lies in the in the region $x_{i+1} \to x_{i-1}$ special treatment is required to handle the jump conditions, as this discretization is only valid in the presence of smooth pressure and density fields. To avoid mixing of values across domains a ghost fluid method is used to explicitly account for the jump conditions at the boundary. The ghost fluid method used for solving these jump conditions is similar to that described by Liu et al. [40], and is summarized in the following two subsections.

Pressure Jump

The first consideration is the pressure jump condition due to surface tension given by Equation 2.45. As the interface has sub-cell resolution the curvature at the interface can be found by interpolating its inverse based on the level set field. Note that this interpolation method is a departure from Liu et al., whose method instead interpolates the curvature directly. While ideally the inverse interpolation would be used in all cases, if the curvature between the two points of interest changes sign, then the calculated result for curvature incorrectly passes through infinity. For this reason the method of Liu et al. as a fallback solution should this occur.

$$\kappa_I \approx \begin{cases} \frac{\kappa^+\kappa^-\left(|\phi^-|+|\phi^+|\right)}{\kappa^+|\phi^+|+\kappa^-|\phi^-|} & \kappa^+\kappa^- > 0 \\ \frac{\kappa^-|\phi^+|+\kappa^+|\phi^-|}{|\phi^-|+|\phi^+|} & otherwise \end{cases} \tag{2.48}$$

Where superscript + and - are values at the next cell on the positive and negative sides of the interface respectively. Figure 2.1 indicates the difference between the two methods.

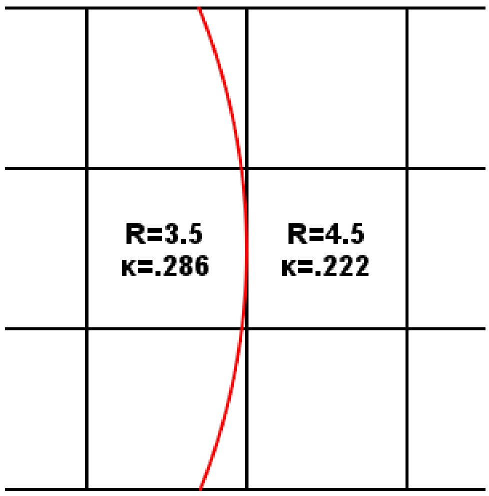

Figure 2.1.: Discretization of a circle with a radius of four grid spacings. Interpolating the radius and inverting the result will give the correct curvature on the interface of 0.250 whereas interpolating the curvature field directly will give a curvature on the interface of 0.254.

It is clear that interpolating from the curvature field will not give as accurate results and that, instead, interpolation of the radius of curvature is preferred. For a convex shape this error effectively increases the effect of surface tension around the entire volume. Although the error in the example above is only approximately 2%, the magnitude of the radius of curvature has potential to be significantly lower than four grid spacings and the error becomes much more significant as the resolution of the curvature decreases.

Defining x_{i+1} and x_i as the positions of cells either side of the interface, the pressure jump due to surface tension can be substituted directly in to Equation 2.47.

$$\frac{1}{(\Delta x)^2}\left(\frac{(p_{i+1}+\sigma\kappa_I)-p_i}{\rho_{i+1/2}}+\frac{p_i-p_{i-1}}{\rho_{i-1/2}}\right)=RHS \tag{2.49}$$

As the pressure jump term is a constant it is convenient to move it to the right hand side of the equation:

$$\frac{1}{(\Delta x)^2}\left(\frac{p_{i+1}-p_i}{\rho_{i+1/2}}+\frac{p_i-p_{i-1}}{\rho_{i-1/2}}\right)=RHS-\frac{\sigma\kappa_I}{(\Delta x)^2\rho_{i+1/2}} \tag{2.50}$$

The other equation featuring a derivative over x_{i+1} and x_i must also be corrected. Once again directly substituting the pressure jump into the equation:

$$\frac{1}{(\Delta x)^2}\left(\frac{p_{i+2}-p_{i+1}}{\rho_{i+3/2}}+\frac{p_{i+1}-(p_i-\sigma\kappa_I)}{\rho_{i+1/2}}\right)=RHS \tag{2.51}$$

and again the constant term can be moved to the right hand side:

$$\frac{1}{(\Delta x)^2}\left(\frac{p_{i+2}-p_{i+1}}{\rho_{i+3/2}}+\frac{p_{i+1}-p_i}{\rho_{i+1/2}}\right)=RHS+\frac{\sigma\kappa_I}{(\Delta x)^2\rho_{i+1/2}} \tag{2.52}$$

It is important to note that the additional value on the right hand sides of Equation 2.50 and Equation 2.52 are the same magnitude with opposite sign. This is expected as they are both correcting for the same jump condition, simply from different sides of the interface. The method for calculating the density term $\rho_{i+1/2}$ will be described below.

Density Jump

Next the density jump condition (Equation 2.46) must be taken into account. Considering once again an interface lying between x_{i+1} and x_i the interpolant θ is calculated:

$$\theta=\frac{|\phi_i|}{|\phi_i|+|\phi_{i+1}|} \tag{2.53}$$

This interpolant can be used to approximate the position of the interface, as it splits the region between the cells into two pieces, of size $\theta\triangle x$ on the left side, and $(1-\theta)\Delta x$ on the right. Although the pressure at the interface is discontinuous the pressure gradient remains continuous when divided by the density. A second order approximation for pressure allows the assumption that this pressure gradient is constant within the cell. This leads to the following approximation:

$$\frac{p_{i+1} - p_\Gamma}{\rho_{i+1}(1-\theta)} \approx \frac{p_\Gamma - p_i}{\rho_i \theta} \approx \frac{p_{i+1} - p_i}{\rho_{i+\frac{1}{2}}} \tag{2.54}$$

Solving Equation 2.54 for the pressure at the interface:

$$p_\Gamma = \frac{p_{i+1}\rho_i\theta + p_i\rho_{i+1}(1-\theta)}{\rho_i\theta + \rho_{i+1}(1-\theta)} \tag{2.55}$$

If an effective density $\hat{\rho}$ is defined:

$$\hat{\rho} = \rho_i\theta + \rho_{i+1}(1-\theta) \tag{2.56}$$

Equation 2.55 simplifies to:

$$p_\Gamma = \frac{1}{\hat{\rho}}\left(\rho_i p_{i+1}\theta + \rho_{i+1} p_i (1-\theta)\right) \tag{2.57}$$

substituting Equation 2.57 back in to the left hand term of Equation 2.54:

$$\begin{aligned}
\frac{p_{i+1} - p_\Gamma}{\rho_{i+1}(1-\theta)} &= \frac{p_{i+1} - \frac{1}{\hat{\rho}}\left(\rho_i p_{i+1}\theta + \rho_{i+1} p_i (1-\theta)\right)}{\rho_{i+1}(1-\theta)} \\
&= \frac{p_{i+1}\left(1 - \frac{1}{\hat{\rho}}(\rho_i\theta)\right)}{\rho_{i+1}(1-\theta)} + \frac{p_i}{\hat{\rho}} \\
&= \frac{1}{\hat{\rho}}\left(\frac{p_{i+1}(\hat{\rho} - \rho_i\theta)}{\hat{\rho} - \rho_i\theta} + p_i\right) \\
&= \frac{1}{\hat{\rho}}(p_{i+1} + p_i)
\end{aligned} \tag{2.58}$$

This analysis shows that the correct inter-cell densities in the presence of a density jump condition with no pressure gradient jump, are given by Equation 2.56. This is an important result. If one were to take inter-cell densities from any other interpolation the jump condition would not be modeled correctly.

2.4.2. Solution of a Sparse Linear System

The pressure projection method outlined above requires the solution of a linear system of equations for $\mathbf{x}$:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.59}$$

where $\mathbf{x}$ is a vector of n unknowns, $\mathbf{A}$ an n by n constant matrix, and $\mathbf{b}$ a vector of n constants. This equation is solved using a preconditioned Bi-Conjugate Gradient Stabilized (BiCGStab) method [41]. The algorithm for this method is documented in Algorithm 2.1.

Algorithm 2.1 Pseudocode for the BiCGStab method

```
$\mathbf{x}_0$ is initialized as a guess of the final solution
$\mathbf{K}^{-1}$ is the inverse of our preconditioning matrix
$\mathbf{w}$ is a weighting factor for each cell
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\tilde{\mathbf{r}}_0$ is chosen as an arbitrary vector not orthogonal to $\mathbf{r}_0$. e.g. $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$
$\rho_0 = \alpha = \omega_0 = 1$
$v_0 = p_0 = 0$
Iterate i=1,2,3,...
    $\rho_i = (\tilde{\mathbf{r}_0}, \mathbf{r}_{i-1})$
    $\beta = \frac{\rho_i \alpha}{\rho_{i-1}\omega_{i-1}}$
    $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta\left(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1}\right)$
    $\mathbf{y} = \mathbf{K}^{-1}\mathbf{p}_i$
    $\mathbf{v}_i = \mathbf{A}\mathbf{y}$
    $\alpha = \frac{\rho_i}{(\tilde{\mathbf{r}_0}, \mathbf{v}_i)}$
    $\mathbf{s} = \mathbf{r}_{i-1} - \alpha\mathbf{v}_i$
    $\mathbf{z} = \mathbf{K}^{-1}\mathbf{s}$
    $\mathbf{t} = \mathbf{A}\mathbf{z}$
    $\omega_i = \frac{\left(\mathbf{K}^{-1}\mathbf{t}, \mathbf{K}^{-1}\mathbf{s}\right)}{\left(\mathbf{K}^{-1}\mathbf{t}, \mathbf{K}^{-1}\mathbf{t}\right)}$
    $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha\mathbf{y} + \omega_i\mathbf{z}$
    if $(mod\,(i, 10) = 0)$ then
        $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$
    otherwise
        $\mathbf{r}_i = \mathbf{s} - \omega_i\mathbf{t}$
    if $\sum \mathbf{r}_i^2 \cdot \mathbf{w} < threshold$
        $\mathbf{x} = \mathbf{x}_i$
        exit
```

Preconditioning

The solver is preconditioned to help accelerate convergence. The aim of a preconditioner is to reduce the condition number (a bound on the inaccuracy of $\mathbf{x}$ after an approximate solution) of $\mathbf{A}$ in Equation 2.59 by per-multiplying both sides by the inverse of a preconditioning matrix $\mathbf{K}$. Typically $\mathbf{K}$ in this case is an approximation to $\mathbf{A}$ for which it is easy to calculate $\mathbf{K}^{-1}\mathbf{y}$. Perhaps the simplest preconditioner is the Jacobi preconditioner where $\mathbf{K}$ is formed by taking the diagonal terms of $\mathbf{A}$ and $\mathbf{K}^{-1}$ is found by simply reciprocating these values.

While the aim of the preconditioner is to increase performance by reducing iteration count, care must be taken not to spend more time calculating the additional matrix-vector products required by the method than time saved due to reduced iterations. Another consideration is that a highly parallel preconditioner must be selected so that it can run effectively on the GPUs. For this reason a simple block-Jacobi preconditioner is used [39], with the blocks being columns spanning the z-dimension.

2.5. Summary

A method of solving immiscible incompressible flows has been presented. The method is capable of solving problems involving viscous flow interactions with surface tension and gravity forces present. The overall method can be subdivided into three smaller methods: the conservative level set method, implicit large eddy simulation and the pressure projection method. While at the start of the project there was an implementation of each method available, they were not compatible, and work was required to combine them. In the process of combining them, several code errors and incompatible assumptions were found and corrected.

Throughout this method the interface is treated as sharply as possible. The one exception to this is the viscosity terms, with the value of fluid viscosity smoothed over the interface. The delta function used to smooth viscosity over the interface is thought to be novel, and ensures that the momenta of the two phases are adversely affected to the same degree by the smoothing.

The method to advance the solution by a discrete time step can be summarized as follows:

1. Calculate the time step Δt from the flow field.
2. For each Runge-Kutta step:
    a) Calculate the level set velocity.
    b) Calculate inviscid, viscous, gravity and level set fluxes and update the flow field based upon these.

   c) Apply the cell merging method to all mixed cells.
   d) Calculate interface parameters such as curvature.

3. Using the pressure projection method enforce the divergence free constraint of incompressible flow fields.
4. Apply reference frame corrections so that the interface remains near the centre of the domain.
5. If required use the level set reconstruction method to ensure the level set field remains a signed distance function.

The method was validated using two test cases: the oscillation of a water droplet under surface tension in the absence of gravity (Section A.1), and the motion of an air bubble rising through a viscous fluid (Section A.2). The rising bubble test case was of particular interest and highlighted a potential problem with the numerical scheme. Instead of the interface being smoothy curved, the edges were noticeably straighter than they should be. At the time it was decided that the cause of this error was imprecise handling of the viscosity terms, and as preliminary tests of water droplets showed no such error, it was considered not to be a problem. Much later a potential mechanism was found in the conservative level set method. As the level set field passes through space, cells are merged and unmerged when the phase volume fraction crosses a threshold. This merging significantly changes the local properties. If interface movements are brought about over a long time by small forces this change might lead to a local oscillation: if a certain cell is mixed the interface is forced in a direction leading the cell to become unmixed. Once it becomes unmixed it then may be forced in a direction leading the cell to become mixed. So long as this oscillation is stable the solution will tend to favour the threshold region, effectively constraining a point. Although the error is small, it is significant.

Having validated the method it was then used to simulate water droplets falling under gravity. The results from those simulations are presented in Chapter 4. The "straightening" error was not noticeable from visualizations of these simulations, possibly due to higher surface tension smoothing out areas of high local curvature, however analysis shows that some results may have been affected by it.

3. Implementation on Graphics Processing Units

Synopsis

To be able to extract best performance out of a technology it is first important to understand the properties of that technology. In this chapter an overview of GPU technology will be presented, along with explanations of how this technology can be used to achieve significantly better performance than CPUs for the problems we are interested in. This performance improvement allows those who wish to run CFD simulations to do so either in more detail, or in a shorter time. Details of implementation and relative performance improvements for each of the numerical methods detailed in Chapter 2 are reported.

Parts of this Chapter have been presented in: J. Appleyard and D. Drikakis. 'Higher-order CFD and interface tracking methods on highly-Parallel MPI and GPU systems'. Computers & Fluids, Volume 46, Issue 1, July 2011, Pages 101–105

3.1. GPU Technology

3.1.1. Background

To understand the advantages of a Graphics Processing Unit (GPU) compared to a Central Processing Unit (CPU) one must first understand the design philosophies around them. Modern GPUs are designed first and foremost to project three dimensional scenes onto a two dimensional plane for computer gaming. These scenes typically comprise hundreds of thousands of polygons, each with effectively arbitrary position and colouring. The GPU must transform these polygons from 3D space to 2D space, sort them by depth, texture them, light them, and finally determine the colour of each pixel so that the full image can be displayed. For a moving scene to appear smooth this entire process must be done in real time, with typical frame rates of at least thirty frames per second.

While this may seem a daunting task it becomes tractable once one key element about all of the above transformations is recognized: almost every operation that has to be performed on one polygon must also be performed on many other polygons. Furthermore these operations are independent. For example, the transformation of a vertex from "world space" to "screen space" is a matrix operation. This matrix operation must be performed on every vertex in the "world" so that its position relative to the virtual camera is known. The result of one vertex transformation does not depend on the result of another so it is possible to do each simultaneously. The same is true to a large degree with the texturing and lighting operations, with the final colour of a polygon usually largely independent of the final colour of another. As it is commonplace for millions of these independent calculations to be required for a single frame the system can be described as *massively parallel*: there's a lot of work to be done and it can all be done simultaneously.

One of the first scientific applications for GPUs was presented by Lengyel et al. in 1990 and concerned robot motion planning [5]. Many other applications have since followed [6, 7, 8, 9], including CFD [42, 43, 44]. At first, the tools were extremely limited and programmers had to treat each parallel operation as analogous to a pixel on the screen, or a vertex on a three dimensional object. For example, in computer graphics the colour of a pixel is often defined by a 32 bit integer, with the red, green, blue and alpha (a value determining how the pixel is blended with those further from the camera) each forming eight bits of this integer. By performing operations such that the colours of these pixels changed in a certain way it is possible to manipulate these integers to perform certain calculations, the answer of which can then be interpreted as the result of a computation. It was not until the release of NVIDIA's Compute Unified Device Architecture (CUDA) programming language in 2007 that scientific computing on GPUs finally became fully decoupled from this graphics model.

3.1.2. Hardware

Once the problem is understood to be massively parallel, it is possible to design hardware to exploit this fact. Perhaps the most obvious design decision that differentiates a CPU and a GPU is the amount of space on the die devoted to processing.

As CPUs are inherently serial it is not necessarily known what data will be called upon for the next calculation. For example, it may be that a process contains a branch which, if followed down one path will require one block of memory, while if followed down another path will require a completely different block. Until the result of the branch is calculated the processor cannot know for sure which block will be required. For this reason CPUs tend to have a large proportion of their die space devoted to caching data that is expected to be required soon, with large multi-level caches to minimize latency should the expected data be required.

GPUs do not need nearly so much cache, and can hence devote more space to processing. As GPUs are designed to deal with large amounts of data in parallel they are made to be most efficient when it is assumed that parallel threads do not depend on each other. This means that while calculations are being processed for one thread, another thread can waiting idle for the memory for its next calculation to load. There is little or no chance that the first thread will invalidate this memory load, either by overwriting the data with new data, or by taking a branch that makes that data unnecessary. As the idle threads can be made to take up very few resources, this method allows the GPU to hide latency due to memory loading, as the GPU is able to load data into memory and perform calculations simultaneously. Because memory latency becomes less important in a massively parallel environment, a GPU die does not need nearly so much space dedicated to caching of memory as a CPU die, and hence can dedicate a lot more space to raw processing power.

This is not the only advantage a GPU has when it comes to performance. As it is known at the outset that the problems the device is to solve will be massively parallel, it is possible to design the hardware to do many calculations simultaneously rather than separately. This means that a GPU can have many simple processing cores rather than a few highly complex ones. While the simplicity means that a single GPU core is less efficient at performing a single series of instructions on a single piece of data than a single CPU core, there are many more simpler cores available. This means that not only does the GPU have more space devoted to processing, but that space can be more efficiently used.

In the early stages of this work a cluster comprising four Tesla C1060s was used to run all the GPU simulations. At a later stage, an improved cluster comprising eight Tesla S2050s was obtained. The basic performance capabilities of each Tesla GPU is listed in Table 3.1.

Device	Multiprocessors	Arithmetic Throughput		Memory Bandwidth
		Single Precision	Double Precision	
C1060	30	933 GFLOPS	78 GFLOPS	102 GB/s
S2050	14	1288 GFLOPS	515 GFLOPS	148 GB/s

Table 3.1.: Performance capabilities of the GPUs used. 1 GFLOPS is equal to one billion floating point operations per second.

It is clear that these devices have a very high arithmetic intensity (ratio of floating operations per second to memory bandwidth). Given that the Tesla S2050s can attain 515 GFLOPS in double precision with only 148 GB/s of global memory bandwidth, for each eight byte floating point number loaded from global memory 28 floating point operations must be completed to maximize floating point throughput. As very few applications have such a high arithmetic intensity most codes are strongly limited in performance by memory access speeds, making efficient use of memory highly desirable.

The two main methods for maximizing memory bandwidth on a GPU are memory re-use and memory coalescing:

Memory re-use To understand how memory can be reused one must first understand the different types of memory available on GPUs. Both the C1060 and S2050 share the same basic three types of memory:

1. Global memory; accessible to every thread on every multiprocessor.
2. Shared memory; accessible to every thread in a single block. Each block resides on a single multiprocessor and the total shared memory for all the blocks on a single multiprocessor is limited.
3. Register memory; accessible to an individual thread only.

Both shared and register memory are very high bandwidth whereas global memory is orders of magnitude lower bandwidth. For this reason, if memory that can be reused is stored in either register or shared memory, it is possible to greatly accelerate the application by reducing usage of the limited global memory bandwidth. One improvement in the S2050 was the inclusion of a small cache for global memory accesses. This cache reduces the requirement for memory reuse in some cases, but has little effect in others.

Memory Coalescing

Memory coalescing allows for the minimum instructions necessary to load data from global memory. Given the highly parallel nature of most GPU applications, the hardware is designed to be most efficient when a small group of consecutively assigned threads reads from consecutive memory locations. This access pattern is known as a coalesced access pattern. If the memory structures can be designed so that this access pattern is obtained then the memory throughput of the application can be increased by up to an order of magnitude over an application with random memory access. The cache on the S2050 mitigates this requirement somewhat if the memory accesses are nearly coalesced, as the additional memory that must be loaded in the case of an uncoalesced access can be stored in the cache.

3.1.3. Software

The GPU implementations presented in this work are based upon NVIDIA's CUDA (Compute Unified Device Architecture) programming model [10]. The CUDA environment comprises extensions to both the C and FORTRAN programming languages allowing the CFD code to be written specifically for NVIDIA GPUs. These extensions allow large numbers of threads to be launched on the GPU from within a program running on the CPU. With a few exceptions, executing the CFD code can be very similar to executing a code written to run on a CPU.

The implementation used for the work outlined here primarily used PGI CUDA FORTRAN, an implementation of CUDA in FORTRAN which also includes PGI's own accelerator language [45].

3.2. Implementation of Discrete CFD Problems on GPUs

The methods described in Chapter 2 were first implemented using traditional CPU methods, before being rewritten for GPUs. The process used to transfer the methods to the GPU was incremental. Each sub-section of the code was considered separately and implemented individually, without the requirement of any other subsection of the code to be running on the GPU. This method allows for significantly easier error-checking as it is possible to isolate certain parts of the code very easily for testing.

The publication: *Higher-order CFD and interface tracking methods on highly-parallel MPI and GPU systems* was written based of the findings from an early part of the project and has been largely reproduced in Section 3.2.1. While the hardware used for this publication is now outdated, many of the findings remain valid as the core technologies remain the same.

After the preliminary study was completed new hardware was acquired. Table 3.2 indicates the hardware used for each section.

	CPU	GPU
Section 3.2.1	Intel Xeon 51xx [46]	S1060
Section 3.2.2 onward	Intel Xeon E5620	C2050

Table 3.2.: Hardware used for each section.

3.2.1. Preliminary Study

The first section of the code to be implemented on the GPU was the level set solver. It was decided that it would act as a good proof of concept and would give a good early indicator of expected performance. The three primary aims of producing this solver were:

1. To develop methods to implement discrete finite-element solvers on the GPU efficiently.
2. To determine what performance improvements could be obtained by using the GPU for CFD methods.
3. To analyze the influence of stencil size on this performance.

At the time the study was preformed the hardware available comprised four Tesla S1060s and an HPC facility based on an 856 processor HP XC Cluster built in 2007 [46], and these hardware configurations are used as a basis for comparison between GPU and CPU technology.

Level Set Solver

The aim of the level set solver is to solve the level set equation (described in Section 2.2.1) on a three dimensional grid. While the level set equation was chosen in this case, the method is fairly generic, and adapting it to most other finite element problems is simple. This is because they all follow the same basic pattern of gathering data surrounding a cell to compute a value at that cell. The method presented here is similar to the method used by both Brandvik and Pullan [47] and Micikevicius [48] and is as follows:

Firstly, the solution space is subdivided into n equally sized cuboidal domains. Each domain spans the solution space in one dimension. The sizes of the other two dimensions are then calculated based on four factors:

1. The size of the solution space in these directions. It is most efficient to have its side length as a factor of the length of the solution space. If this is not the case additional logic is required to prevent threads outside of the domain from executing.
2. The limitations on the availability of the fast shared memory and register spaces on the GPU. Larger cuboids require more shared memory. It is also more efficient to have a cross-section as close to square as possible so as to minimize boundary data.
3. Memory coalescing requirements. Memory can be coalesced if the length of one side is a multiple four in double precision.
4. Number of shared memory bank conflicts. If one side length is a multiple of 16 then shared memory is guaranteed to be accessed in the fastest possible way (conflict free). Side lengths which are not multiples of 16 may still access shared memory conflict free, however it is not guaranteed in every case.

The optimum size of the cross-section varies with problem size and the order of accuracy required, however, is typically 8×16 or 16×16. Larger cross-sections require too much memory while smaller cross-sections are inefficient.

Having subdivided the solution space a thread block is assigned to each of the cuboidal domains. Every time step the thread blocks iterate in parallel through the solutions space spanning dimension using shared memory and registers to explicitly cache data required for the next iterations. Each thread in the thread block calculates the result for a single cell per iteration. This is shown diagrammatically in Figure 3.1. Shared memory is used to swap data between the threads of the thread block so as to minimize loading from global memory. If the limitations on shared/register memory availability were lifted this method would allow for each global memory location to be read from only once. Instead, each thread block must load data from neighbouring domains, decreasing efficiency.

A more naïve solution would be to simply assign each thread to perform the calculations for one grid cell. While this is a lot simpler (as it requires no shared memory programming) it would result in an order of magnitude increase in global memory requirements due to the lack of memory re-use. This would therefore be a lot slower.

If multiple GPUs are being used boundary data must be transferred between GPUs after each iteration. Due to hardware limitations there is no way of directly transferring data between GPUs and so the data must be copied across the PCI-E bus to the host memory before being copied onto a different GPU. The maximum theoretical bandwidth of this transfer is 8 GB/s (an order of magnitude slower than GPU global memory). Fortunately, it is possible to copy data across the PCI-E bus while continuing calculations on the GPU by splitting the algorithm

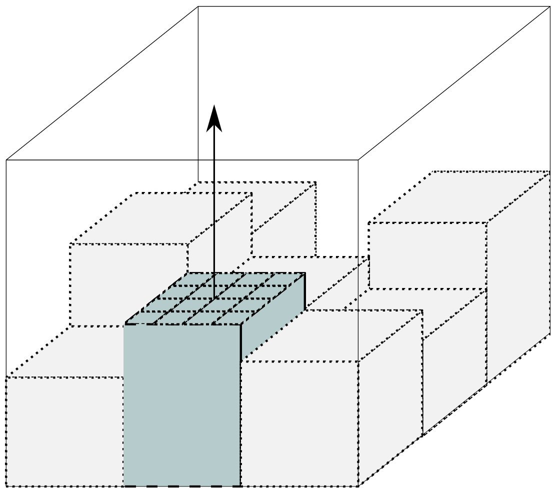

Figure 3.1.: Thread blocks spanning the domain in two dimensions iterating in parallel along the third dimension.

into two sections (one which requires the boundary information and one which does not). This effectively hides the memory transfer with only a small cost due to the splitting.

The CPU implementation is much simpler than the GPU implementation. Each core iterates over a small part of the domain before transferring data between cores. The same asynchronous memory transfer masking technique as in the GPU method is used.

Performance

The test case used to generate these results was the motion of a slotted sphere in a rotational velocity field. The grid was strongly scaled across many devices, i.e., total data processed remains constant. Extrapolation boundary conditions were used at each of the interfaces. It should be noted that while the velocity field was constant in time it was treated as a variable and no optimization was based upon it being constant.

The first set of results are a simply architectural comparison between the GPU systems and the CPU system. Figure 3.2 shows the arithmetic throughput of the CPU and C1060 architectures using a 3rd-order HOUC scheme in single precision. As each architecture solves the same equations this can be used as a direct measure of performance.

Table 3.3 shows the equivalent processing power of multiple C1060s in terms of CPU cores. It is clear to see that a single GPU is capable of performing two orders of magnitude more work than a single CPU core when solving the level

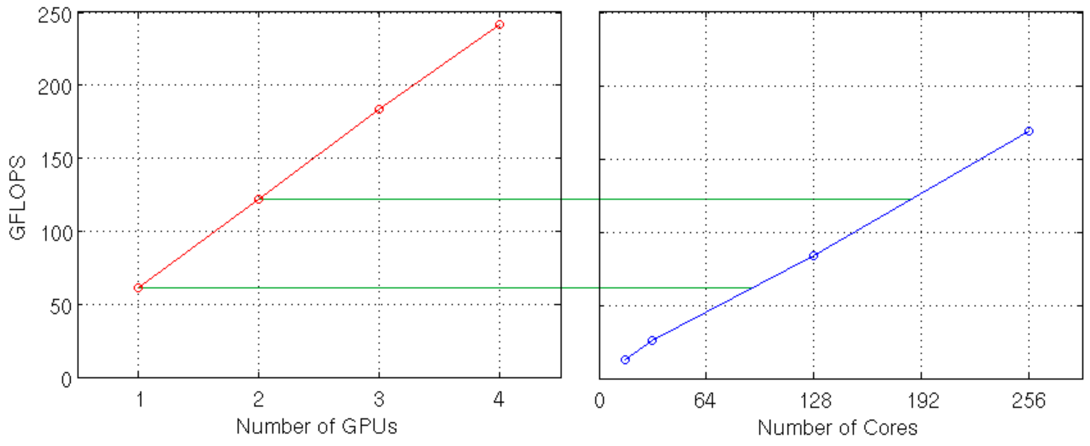

Figure 3.2.: GFLOPS produced from a 3rd-order HOUC scheme in single precision by up to 4 C1060s (left) and by up to 256 CPU cores (right).

set equation. It is also apparent that both architectures scale in performance in a close to linear manner.

C1060s	1	2	3	4
Equivalent Cores	92	187	280	371

Table 3.3.: Equivalent single-precision CPU cores for up to 4 C1060s.

Order of accuracy

Figure 3.3 shows the arithmetic throughput of the CPU and C1060 architectures using 3rd, 5th, 7th and 9th-order HOUC schemes in single precision.

Figure 3.4 shows the arithmetic throughput of the two architectures using 3rd and 9th-order HOUC schemes in single and double precision. Figure 3.5 shows the memory bandwidth required on the two architectures using the same configurations.

As is to be expected the double precision throughput on the GPU is significantly lower than the single precision throughput, however, it is a much greater proportion (30–50%) of the peak theoretical throughput. Comparatively, the CPU shows approximately half the throughput in double precision than in single precision. This suggests that the CPU method is bound by the memory bandwidth of the system. Comparison of the memory bandwidth achieved to that achieved on the same system by the STREAM [49] benchmark confirms this.

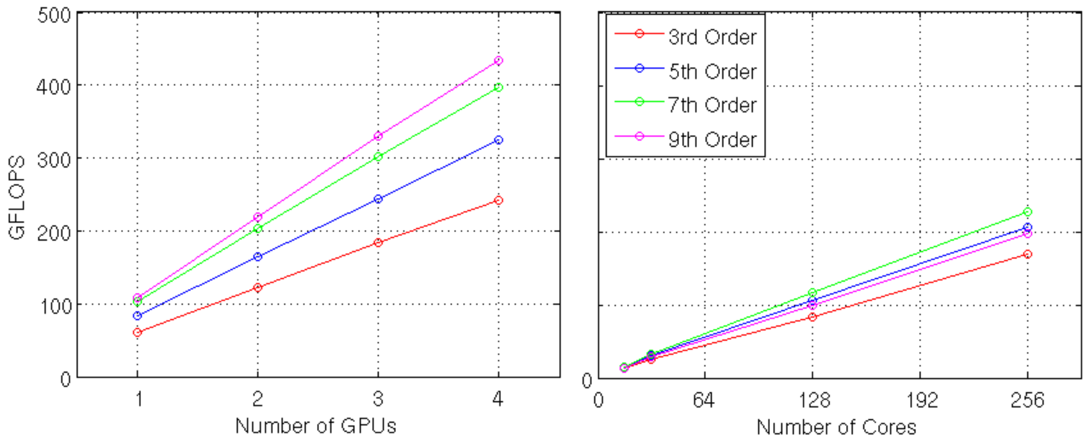

Figure 3.3.: GFLOPS produced from 3rd, 5th, 7th and 9th-order HOUC schemes in single precision by up to 4 C1060s (left) and by up to 256 CPU cores (right).

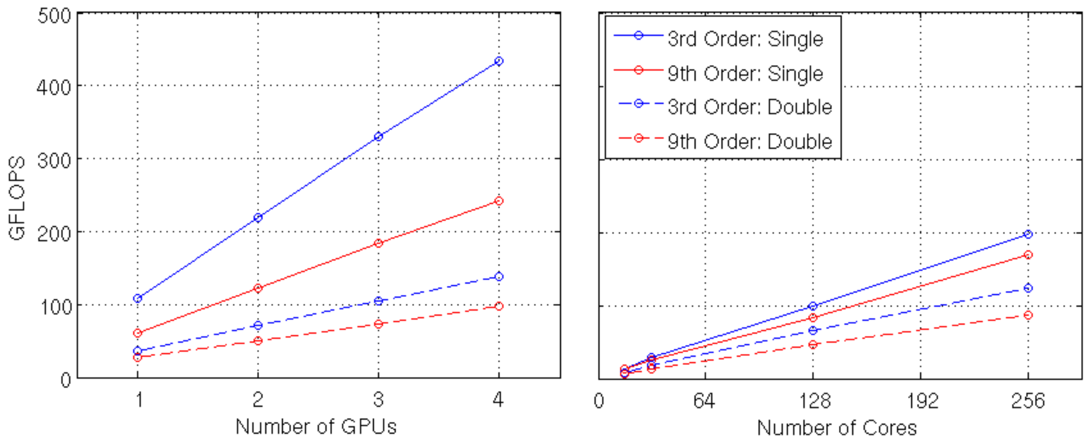

Figure 3.4.: GFLOPS produced from 3rd and 9th-order HOUC schemes in single and double precision by up to 4 GPUs (left) and by up to 256 cores (right).

Conclusions

These data show that the GPU method is bound solely by neither the two constraints (memory bandwidth or arithmetic throughput) of the GPU. Instead, it would appear that the algorithm is bound by both at different stages of execution. This is possible despite the massively parallel nature of the GPU as each iteration requires two steps and after each a block-wide synchronization occurs. The first step mainly comprises memory transfer while the second mainly comprises arithmetic operations. As a multiprocessor typically executes only two blocks concurrently, the system is prone to bottle-necking in either step.

The proportion of time that the GPU spends bound by each limit varies with precision and order of accuracy. To illustrate this: although in both precisions

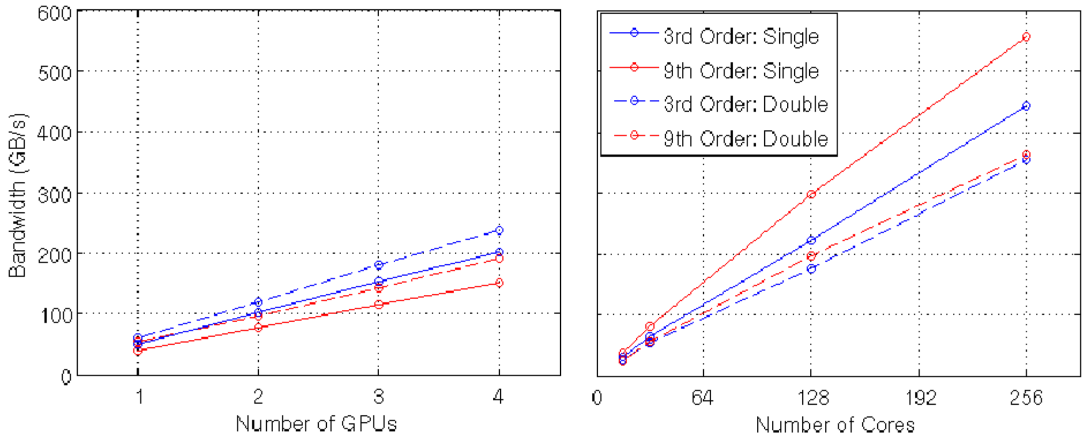

Figure 3.5.: Memory bandwidth for 3rd and 9th-order HOUC schemes in single and double precision by up to 4 GPUs (left) and by up to 256 cores (right).

the 9th-order scheme shows improvements over the 3rd-order scheme, in double precision this improvement is nowhere near as significant. This is due to the inferior double precision performance of the GPU leading to a greater proportion of the execution time spent doing arithmetic operations. This effect is not seen so prominently in single precision as increasing the order of accuracy does not bring the arithmetic throughput to such a large fraction of the peak.

It should be noted that a lot of these results were collected quite early in the project, and although they remain valid as a guideline, improvements in technology have naturally brought about improvements in performance. It was found that the newer S2050s performed approximately 50% faster than the C1060s in double precision calculations. The exact performance improvement varied with case, but not significantly. This indicates that while arithmetic throughput was found to be a partial constraint for the older C1060s, the vastly improved throughput of the S2050s shifted the primary constraint to global memory bandwidth.

3.2.2. Momentum Fluxes

Both viscous and inviscid momentum fluxes were implemented using a very similar method to the level set solver (Section 3.2.1). Fundamentally the process is the same: iterate through a domain in columns using only data from nearby cells. While the actual calculations within each cell are significantly different the “gather and calculate” pattern is the same.

There are however some notable differences between the two which don’t allow the momentum fluxes to be nearly as efficient on the GPU relative to the CPU. Firstly, each cell requires significantly more information and outputs significantly more data. Not only must momenta in all three dimensions be gathered from the entire stencil rather than just a single velocity value, but other information such as aperture sizes must be loaded for each cell. Secondly, the momentum solves must iterate three times through the domain, once along each Cartesian axis. This not only impacts on memory coalescing, but also on overall efficiency. Rather than load each value once, it must be loaded thrice. This adds significantly to the memory bandwidth requirements of the GPU. Finally, while it would be desirable to have all of the data stored on the GPU memory global memory size restrictions often do not allow for this. As the S2050s have only 3GB of global memory, and there is significantly more data to be stored than for the level set solver, this was a significant issue. While it was found that a lot of data could remain on the GPU some moderately sized copies to and from the GPU were required during the calculation of the fluxes to minimize the global memory footprint of the code.

Table 3.4 indicates the relative performance of a single S2050 compared to a single CPU core. These data are compiled over a complete set of three fluxes (inviscid, viscous and level set) combined, along with associated copies to and from the GPU.

Domain Size	Runtime (ms)		Speed-up
	CPU	GPU	
32^3	47.1	16.7	2.8
64^3	496	67.9	7.3
96^3	1693	211	8.0

Table 3.4.: Relative performance of the CPU and GPU for computing momentum and level set fluxes.

While these results initially may seem disappointing compared to the preliminary results it will be shown later that they do not contribute hugely to the overall run time. It may be that with additional work it would be possible to accelerate the GPU runtime; however this was not decided to be a priority.

3.2.3. Velocity Extrapolation

The interface velocity extrapolation method described in Section 2.2.1 is quite expensive in terms of run-time. As the procedure must be run once every Runge-Kutta iteration, and must undergo a reasonable number of iterations on all cells in the domain to reach convergence, there is a significant amount of work to be done for this seemingly simple task.

The GPU implementation of this method is simple yet effective. The odd iterations of the algorithm read the upstream values from one array into each cell, calculates the new value for that cell, and writes it to a second array. This second array is then used as the input for the even iterations, with the first array for the output. Shared memory was used to reduce the number of global memory loads by loading small cuboid volumes into shared memory at the start of each iteration, and then loading the data from the shared instead of from global memory. While this method reduces the required memory bandwidth of the problem by approximately a factor of two, it was found that the impact on performance was much less significant than this. This is likely due to the global memory cache of the S2050 filling a similar role to the shared memory without need for hand-coding.

Table 3.5 indicates the relative performance of a single iteration of the velocity extrapolation implementation on a single S2050 compared to a single CPU core.

Domain Size	Runtime (ms)		Speed-up
	CPU	GPU	
32^3	0.69	0.27	2.6
64^3	4.43	0.48	9.2
96^3	14.2	1.27	11.2

Table 3.5.: Relative performance of the CPU and GPU for extrapolating level set velocities.

3.3. Implementation of Linear Solvers on GPUs

One of the more computationally expensive parts of each time step is the calculation of the pressure field required for the Pressure Projection method. The method used to calculate this field is outlined in Section 2.4.2. Each iteration of this solver requires many level 1 Basic Linear Algebra Subprograms (BLAS) [50] operations, as well as at least five large sparse matrix-vector multiplications. The preconditioning algorithm also requires the solution of many independent tri-diagonal systems.

3.3.1. Level 1 BLAS Operations

There are many options for implementing the level 1 BLAS operations required, including NVIDIA's own cuBLAS library [51] however it was decided for the sake of simplicity to implement these using the PGI Accelerator [45]. While the performance of the PGI Accelerator may not be optimal compared to hand-coding the kernels, profiling showed that these operations weren't a significant proportion of runtime. The implementation of these within the Accelerator was trivial and in most cases could be achieved by simply wrapping the relevant CPU code in preprocessor directives. This has the advantage of making the code easier to understand quickly without significantly impairing performance.

3.3.2. Sparse Matrix-Vector Multiplications

Given the matrix to be solved contains all of its elements within diagonal runs, each diagonal is stored in its own data structure. This method is both simple, compact, and provides optimal coalescing for the matrix-vector multiplication operation. As this particular operation is the most significant in terms of performance due to its high memory bandwidth requirements, it is important to optimize memory structures around it. As each element of the vector must be loaded multiple times in this subprogram, the vector was explicitly loaded into shared memory blocks 256 elements wide. Use of these shared memory blocks resulted in a decrease in global memory bandwidth required by a factor of approximately two, significantly increasing performance.

3.3.3. Tri-diagonal Solver

The linear solver is preconditioned with a block-Jacobi preconditioner. This block-Jacobi preconditioner requires the solution of a tridiagonal matrix which can be subdivided into n smaller tridiagonal sub-matrices each of which is formed by the near-diagonal coefficients of a single line of cells spanning the domain. Each sub-matrix is solved in parallel using the Thomas algorithm [52]. For maximum efficiency some pre-processing is done in the first iteration of the solver to reduce work in all future iterations at the cost of a small amount of memory.

Although the data structures used to store the coefficients are designed for use in the sparse-matrix vector multiplication algorithm, they also allow for very good coalescing in the Thomas algorithm implementation, and hence the tri-diagonal matrices can be solved very efficiently.

3.3.4. Performance

Table 3.6 shows the difference in performance between a single core of a CPU and a single GPU. The results clearly show that while the speed-up is significant at small domain sizes it becomes very beneficial on larger domains.

It should be noted that on the CPU, much more sophisticated preconditioners can be implemented efficiently as there is no longer the restriction that they must be massively parallel. These preconditions have not been implemented here and

Matrix Size	Single iteration time (ms)		Speed-up
	CPU	GPU	
32^3	3.99	0.38	10.4
64^3	35.2	1.52	23.1
96^3	118	4.61	25.6

Table 3.6.: Relative performance of the CPU and GPU based matrix solver.

as the most efficient preconditioners for linear systems often are very serial, the comparison is somewhat unfair to the CPU. The scale to which it is unfair is outside the scope of this work; however a related work co-authored by the author of this work suggests that the improved convergence on serial hardware is rarely enough to compensate for gains from the massive parallelism of GPUs [53].

3.4. Performance Summary

The full numerical method was found execute approximately nine times faster on a single GPU than on a single CPU core. This performance is broken down by part in Table 3.7.

Method	Step time (s)		Percentage runtime		Speed-up
	CPU	GPU	CPU	GPU	
Momentum and level set Fluxes	7.3	0.68	10%	8%	10.7
Velocity Extrapolation	8.4	0.96	11%	12%	8.8
Pressure Projection	55	4.7	73%	59%	11.7
Other	4.5	1.7	6%	21%	2.6
Total	75	8.0	-	-	9.4

Table 3.7.: Performance summary. The results were averaged over a long run typical of the intended use. The domain size was 96 by 96 by 132 cells. The "other" column represents many small methods, each of which was too small to warrant separate accounting. These include volume fraction and aperture calculations, normal and curvature calculations, reinitialization, cell mixing and output.

The most important result here is that the pressure projection process takes the majority of the time on both pieces of hardware. If any further work was to be done on optimization, further study of the runtime of this section seems the obvious place to start. While the BiCGStab method was shown to be greatly accelerated by implementation on the GPU it would appear that the methods surrounding it are perhaps not as fast as they could be. Another section which may require further attention for peak performance is the *other* section. This section includes many small methods which combine to take a significant proportion of run-time. Although some of these methods are running on the GPU, others are not, and it may be that some extra performance can be found here.

It is noteworthy that the velocity extrapolation method is not performing as well as it was found to in Section 3.2.3. Although the difference is only a few percentage points, the change in domain size or iteration count is not expected to lead to worse performance. Instead, this change can probably be attributed to the change in shape of the isosurface over time. As the extrapolation method updates only a subset of points in its early iterations, an alteration to this subset can result in a change in the requirements of the algorithm. A smaller subset will tend to under-occupy the GPU, hence reducing relative GPU performance.

While the GPU has been used to accelerate the program significantly compared to a single core, the overall speedup is not as high as had been hoped based upon the preliminary investigation, with the overall speedup being approximately a factor of 10. The poor performance is probably due to the additional complexity of the full code. One problem is that the comparison between CPUs and GPUs is hard to do fairly. While here only performance is compared, age, cost, and power consumption are also important considerations. In the preliminary investigation the GPUs were newer than the CPUs, whereas in the full code the comparison is between hardware with similar release dates.

One unexpected advantage of the GPU was the increased size of domain that could be executed on a single GPU node in a reasonable execution time. The ability to execute simulations on a single node is a significant advantage, because some of the methods used don't perform particularly well in a multi-node environment (specifically, neither the velocity extrapolation nor the BiCGStab method scale particularly well: the former often only updates a subset of cells, while the latter requires very frequent inter-node communication). When conducting numerical simulations many different input parameters could be simulated simultaneously, each using a single GPU, giving a lower overall runtime than if they were scheduled sequentially. This would not be possible using CPUs as each simulation would take an order of magnitude longer, and instead multi-node simulations would be required with their associated loss in efficiency. For all of these reasons, it's unfair to concentrate on a single metric, such as speed-up value, when comparing CPUs and GPUs.

4. Water Droplets Under Gravity

Synopsis

In this chapter the 3D flow solver described in the previous two chapters is used to simulate water droplets falling through air under gravity. There have been a large number of experimental and theoretical investigations into the the properties of this system. The two main properties of interest in these investigations are the shape and velocity of droplets at terminal velocity. The aim of the work described in this chapter is both to reproduce experimental results, and to provide new insight into these properties. By reproducing results for water falling through air it is then possible to say, with reasonable confidence, that properties of similar fluid pairings acting under similar forcing can be simulated accurately.

It is planned that parts of this chapter are to be presented in: **J. Appleyard** and D. Drikakis. 'CFD Investigation into the properties of water droplets falling through air'

4.1. Introduction

In Chapter 2 numerical methods designed to simulate incompressible, immiscible multi-phase flow were outlined. Appendix A describes some validation tests used to confirm that these numerical methods are indeed working as intended. In the current chapter these methods are used, having been implemented on GPUs as described in Chapter 3, to simulate water droplets of various sizes falling freely through undisturbed air. The results from these simulations will be compared to experimental results with the aim of not only validating the numerical methods, but also providing new insight into single droplet dynamics. With these methods validated, it is hoped that future work into related fields, such as the breakup of large droplets or the modeling of arbitrary fluids in similar situations should be possible with simple extensions to the code.

The falling water droplets have been chosen to have similar properties to single droplets in naturally occurring rain at sea level. The study of raindrops is interesting from an engineering standpoint and from a standpoint of pure curiosity. As a natural phenomenon rainfall is something which is experienced by everybody, yet its properties are often misunderstood. The primary engineering application is in improving the ability of radar to determine the characteristics of rainfall for meteorological purposes [54].

Studies into the terminal velocity of falling water droplets primarily took place in the first half of the 20th century [55, 56, 57, 58], while studies into drop deformation at these terminal velocities have mostly taken part in the second half of the 20th century. Notable research was carried out in wind tunnels by [59, 60], and mathematical models describing describing the equilibrium shape of water droplets at terminal velocity have been proposed [61, 12]. In conducting these experiments it was noticed that the droplets under study tended to oscillate as they fell, and the parametrization of these oscillations became the next focus of study in both laboratory conditions [62, 63, 64, 65, 66], and in the field [67, 68, 69, 70]. Both Beard et al. and Szakall et al. have recently summarized much of the experimental and theoretical work on drop shapes and oscillations to date [13, 71].

In the field of CFD several authors have studied axisymmetric deformations of liquid droplets [72, 73, 74], however the axisymmetric assumption has been shown both experimentally and theoretically to break down for larger droplets [74]. Other CFD work on droplets make the assumption that the effects of the air phase is negligible. For example, the work of Lycett-Brown and Luo[75] uses the Lattice Boltzmann Method to study droplet collision, in a regime where the ratio of surface tension to inertia of a droplet is higher than the droplets this work studies. In this case, the assumption of no deformation due to air is reasonable[12].

Neither of these assumptions can be made when studying the full case, as the deformations of the droplets are expected to be driven by air flow around each droplet and these deformations are not expected to be axisymmetric. As such, the simulations presented in this chapter will be the first of their kind.

The droplet sizes studied will range in size from an equivalent spherical radius of 1mm to 3mm. The main results that this work aims to reproduce from experiment are:

1. The terminal velocity of the droplets. At the lower end of the size range the terminal velocity is expected to be approximately that of an equal mass sphere while at the upper end it is expected to be significantly lower.

2. The mean and transient axis ratios the droplets, as well as their mean and transient profiles. At the smaller end of sizes the mean ratio between vertical and horizontal axis length of the droplets is expected to be approximately 0.9, while at the larger end it is expected to to be approximately 0.65. This axis ratio is expected to oscillate significantly and these oscillations are to be characterized.

3. The internal circulation within the droplets. The pressure and viscosity forces acting on the droplet should cause circulation within the water. Given the density of water is so much greater than that of air it is expected that this circulation will be significantly slower than the external flow.

Further to this, the properties of the wake of the droplets is to be examined. It is expected that falling droplets will shed eddies, much as a sphere would at similar Reynolds numbers. There is very little experimental data available for these properties, with the work of Saylor and Jones as the only reference studying relevant droplet sizes [76].

Section 4.2 will describe the simulation conditions for all simulations. Each subsection in Section 4.3 will describe the current theory, any relevant experimental results, and present the results of the simulations for one of the following properties of the droplets:

- Terminal velocity.
- Mean profile.
- Droplet oscillations.
- Turbulent wake.
- Internal circulation.

The results findings will then be summarized in Section 4.4.

4.2. Parametrization

The physical properties of the system were chosen to match the properties in the terminal velocity experiments of Gunn and Kinzer as closely as possible [58]. These properties are tabulated in Table 4.1.

	Water	Air
Density ($kg \cdot m^{-3}$)	998	1.20
Dynamic Viscosity ($\mu Pa \cdot s$)	1002	18.3
Surface Tension ($mN \cdot s^{-1}$)	72.9	
Gravity ($m \cdot s^{-2}$)	9.81	

Table 4.1.: Physical properties of the system.

Droplet radii (r) between $1mm$ and $3mm$ were simulated. The domain size was chosen to be proportional to the droplet radius and had dimensions$(20r, 20r, 33r)$, with the droplet initialized centred at $(10r, 10r, 10r)$. The domain was discretized using a rectilinear grid with a core uniform grid of dimension $(4.5r, 4.5r, 5.625r)$. A co-ordinate system was defined such that the droplet centre was at the origin. Gravity was defined to act along the negative z-axis. Spatially constant velocity boundary conditions are enforced on all faces of the domain. The grid size for the domain was $(96, 96, 132)$, with a core of $(48, 48, 60)$, for all cases. The resultant core grid spacing was $\frac{3r}{32}$, giving approximately 21 grid cells per dimension for a spherical droplet.

Each droplet was initialized in the equilibrium shape described by the Beard and Chuang model [12]. The water was initialized at rest, with air moving at uniform velocity filling the rest of the domain. The initial velocity of the air was taken to be 90% of the expected terminal velocity. Once the terminal velocity was found to vary by no more than $0.02 \,{}^{\mathrm{m}}\!/\!{}_{\mathrm{s}}$ over a 1m fall distance, and there were no indications of any influence of initial conditions on droplet oscillations, data on wake and drop shape was logged.

4.3. Properties of Freely Falling Water Droplets

4.3.1. Terminal Velocity

The terminal velocity of water droplets through stagnant air at sea level was established experimentally first by Laws, and then by Gunn and Kinzer [57, 58]. As both authors report similar results, and no more recent work has shown these results to be significantly in error, they will be assumed to be an accurate basis for comparison. Figure 4.1 shows the terminal velocities of the simulated droplets.

For comparison the results of Gunn and Kinzer and results for a rigid sphere [77] are also shown.

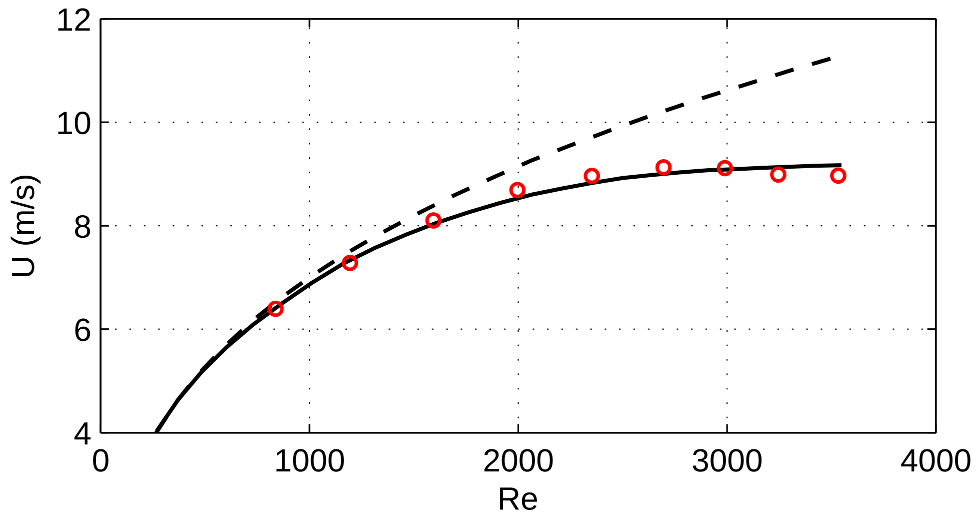

Figure 4.1.: Terminal velocity variation with droplet Reynolds number. Simulated results are represented by circles. The solid line represents the experimental results of Gunn and Kinzer. The dashed line represents the results for a solid sphere.

The terminal velocities from simulation correspond well with the results of Gunn and Kinzer, with the largest error being approximately 3%. The experimental error bounds set by Gunn and Kinzer was 0.993%. It is unclear whether this error is due to inaccuracies within the numerical model, or whether the simulation simply required a little more time to converge on the experimental result. Alternatively, the simulated conditions may have been slightly different to the conditions studied by Gunn and Kinzer.

Although the droplets principle direction of motion was aligned with gravity, there were horizontal velocities observed in some cases. These drift velocities have been noticed in experimental studies, with drift velocities of up to 30% of the terminal velocity recorded [78, 69]. However in the simulations the magnitude of the drift velocity measured was, at greatest, only 2% of the vertical velocity. These drift velocities are not fully understood, but are thought to be related to asymmetric vortex shedding.

One possible cause of error could therefore be the initialization of the simulated drops at speed. It has been reported that, for spheres, asymmetric wakes are produced at a Reynolds numbers between 300 and 420 [79]. These asymmetric wakes are expected to produce asymmetric drag, and therefore induce drift velocities. By not simulating the droplets as they pass through this Reynolds number range, lateral drift is erroneously not induced as the droplet accelerates.

4.3.2. Mean Profile

Theory and Experimental

The two main models describing the equilibrium profile of droplets are the perturbation model of Pruppacher and Pitter [61], and the force balance model of Beard and Chuang [12]. Of these two models the force balance model appears to give more accurate correlation with experimental data [13]. To produce this model it was assumed that internal pressure remained hydrostatic (ie. no internal circulation) and the external pressure was based upon distortions of the time-averaged pressure distribution around a sphere. While it's clear that these assumptions will not be valid for the unsteady profile of a droplet they can be used to provide a baseline for the "average" shape of droplets.

Simulation

Figure 4.2 compares the axis ratios calculated using the force balance model with the mean axis ratios calculated through simulation. While most droplets correspond to the model, the $2mm$ radius droplet shows significant deviation. While the calculated terminal velocity for this droplet is slightly larger than the expected terminal velocity (and therefore a slightly smaller axis ratio is expected), this seems insufficient to account for the difference, and the $2mm$ radius droplet terminal velocity is not significantly more in error than other droplets. Further investigation is needed to determine the cause of this discrepancy.

Although droplets were found to oscillate (Section 4.3.3), the Beard and Chuang model was found to be a good predictor of their approximate shape. Figure 4.3 compares various simulated drops appearing at their theorized equilibrium state.

4.3.3. Droplet Oscillations

Theory and Experimental

It has been found that water droplets of diameter of $1mm$ or greater falling at terminal velocity through air oscillate as they fall [63, 64, 65]. A first approximation to the natural frequencies of these oscillations can be calculated by assuming that the droplets remain spherical. With this simplification the natural frequencies determined by Rayleigh should be correct for small oscillations [80]. For a given spherical drop of radius r it is the natural frequencies are given by Equation 4.1.

$$\omega_0^R = \sqrt{\frac{\sigma n(n-1)(n+2)}{\rho r^3}} \tag{4.1}$$

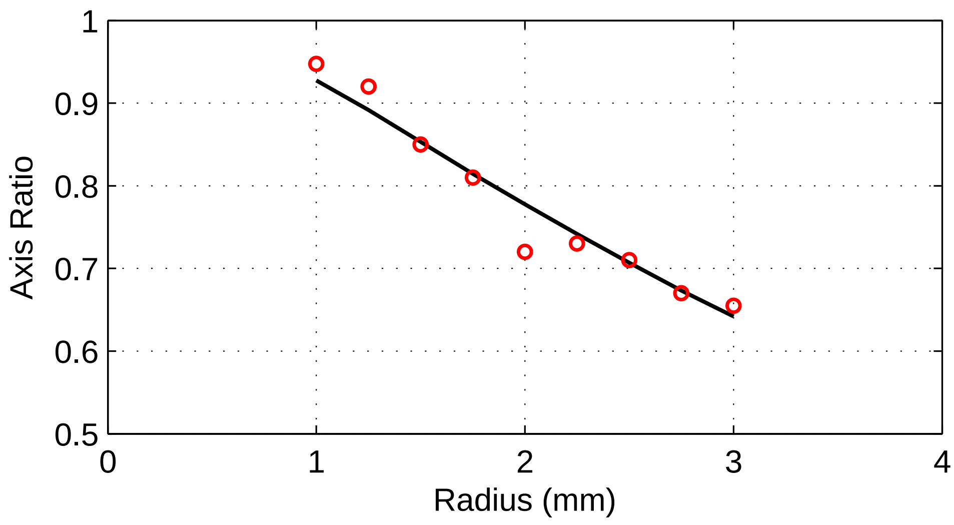

Figure 4.2.: Mean axis ratios for 1-3mm radius droplets. The circles indicate simulated results. The solid line shows the equilibrium results from the Beard and Chuang model [12].

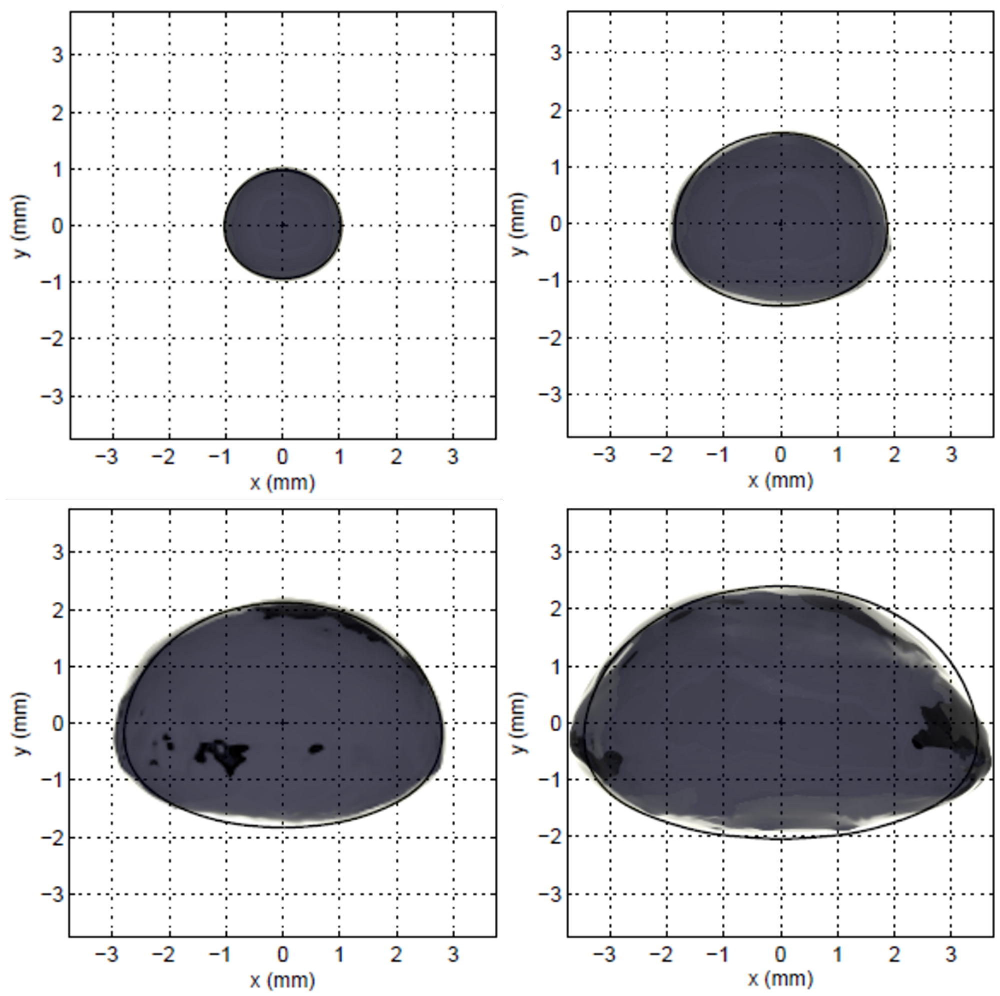

Figure 4.3.: Simulated drop profiles (filled area) compared to the Beard and Chuang model (solid line). The droplet radii illustrated are 1mm (top left), 1.75mm (top right), 2.5mm (bottom left) and 3mm (bottom right). Droplets were found to oscillate and therefore the profiles do not quite match the equilibrium profiles.

The first non-trivial natural frequency is found by substituting $n = 2$:

$$\omega_{0,2}^{R} = \sqrt{\frac{8\sigma}{\rho r^3}} \tag{4.2}$$

The modes at each natural frequency are expected to have n nodes.

This model is acceptable for a droplet oscillations at rest, however it is not sufficient if the droplet is affected by strongly directional external forces, such as gravity. As a result of this imposed directionality each value of n produces $m = n+1$ distinct modes instead of simply one [62]. The characteristic frequency of a given mode, m, is approximated by:

$$\omega_{0,n,m} \approx \omega_{0,n}^{R} \left(1 - \frac{A_0^{(2,1)}(n,m)}{4n(n-1)(n+2)} \hat{u}^2\right) \tag{4.3}$$

Where $\hat{u}$ is a non dimensional flow velocity defined by $\hat{u} = (0.34447 \cdot We_r)^{0.5}$. For the first non-trivial value of n (ie. $n = 2$) the frequency modification factor $\frac{A_0^{(2,1)}(2,m)}{4n(n-1)(n+2)}$ takes values of -0.00804, 0.0241 and 0.121 for $m = 1, 2, 3$ respectively [62]. These three modes have been classified by Beard et al. as axisymmetric ($m = 1$), transverse ($m = 2$) and horizontal modes ($m = 3$) [13]. Figure 4.4 shows the approximate form for spherical droplets of these modes pictorially. For droplets which have equilibrium profiles significantly distorted from spherical these shapes are very much indications of the approximate shape of the drop distortions rather than exact descriptions.

The three distinct low-frequency oscillation modes have the potential to interact with the turbulent wake of the droplet, if their natural frequencies are found to be similar to the eddy shedding frequency of the droplet. By comparing the natural frequencies of the system to the eddy shedding frequencies of a sphere, and confirming this result by experiment, Beard and Kubesh show that the frequencies at which eddy shedding occurs are too high to directly resonate with droplets significantly larger than 0.8mm radius [64]. In a later work by the same authors it is suggested that *subharmonic* resonances are possible, if the eddy shedding frequency is an integer multiple of the natural frequency [81]. Section 4.3.4 will examine the wake of the droplets in an attempt to determine how the wake could interact with the droplet oscillations.

An alternative method of forcing may come about through *drag forcing* [82, 81]. A drop becoming more oblate (ie. flattening) will tend to have an increase in

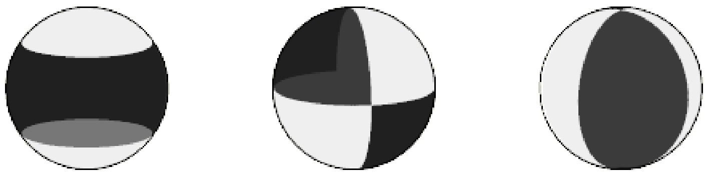

Figure 4.4.: $n = 2$ modes for axissymetric (left), transverse (centre) and horizontal (right) modes. The black and white regions depict amplitudes of opposite sign, with their edges representing the anti-nodal lines[63]. The gravity vector acts vertically.

pressure at its upstream surface. This increase in pressure will lead to further flattening, and hence further pressure. The flattening, however, will increase the frontal area of the droplet, leading to higher overall drag. This drag will cause the drop to decelerate, stabilizing the positive feedback due to decreased pressure at lower speeds. Depending on the time scales of the flattening and deceleration response, it is possible that drag forcing could create a larger response than forcing via eddy shedding.

While the frequencies of the modes of droplet oscillation are well understood, the expected amplitudes of the oscillations at each mode are not. Both Szakáll et al. and Beard et al. report a second order polynomial fit from empirical data [71, 13]. Szakáll et al. giving the following equation for the peak-to-peak amplitude of the axis ratio A_α [71]:

$$A_\alpha = 3.6 \cdot 10^{-3} D_0^2 + 2.13 \cdot 10^{-2} D_0 \tag{4.4}$$

Where D_0 is the equivalent sphere diameter with units of mm.

Equation Equation 4.3 can also be used to determine the expected natural frequencies of higher order modes. For the four oscillations for which $n = 3$, the coefficient $\frac{A_0^{(2,1)}(3,m)}{4n(n-1)(n+2)}$ takes the values -0.00357, 0.0107, 0.0536 and 0.125 for modes $m = 1, 2, 3, 4$ [62].

As with the $n = 2$ modes the approximate mode shapes can be represented pictorially Figure 4.5.

There has been a large amount of experimental work conducted examining oscillations of falling droplets, both in wind tunnels and in the atmosphere [62, 63, 64, 81, 66, 83]. For a review the reader is directed to the summary of Beard et al. [13].

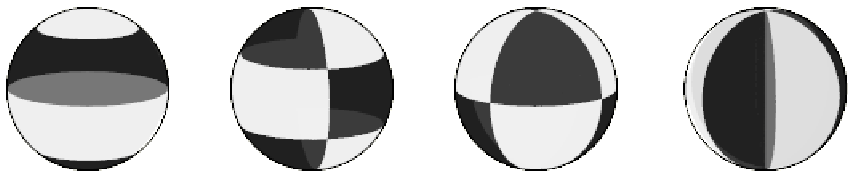

Figure 4.5.: $n = 3$ modes for axissymetric (left), transverse (centre-left and centre-right) and horizontal (right) modes. The black and white regions depict amplitudes of opposite sign, with their edges representing the anti-nodal lines[63]. The gravity vector acts vertically.

Simulations

There are several properties of the fundamental modes to be reproduced from experiment. By computing the power spectral density of the axis ratio signal, it is possible to determine with reasonably high accuracy the frequencies of the fundamental modes. The power spectral densities for each droplet are presented in Appendix B, and the extracted frequencies are compared with theory in Figure 4.6. While these data follow the general trend expected from theory, the fit is not as good as experimental data [13]. It is certainly not possible to distinguish between different modes by their frequency alone. Part of the difficulty is the lack of resolution in the power spectral density, indicating that the simulation time was too short.

It should be noted that there is a slight flaw in this methodology which was noticed too late to be corrected: it is possible that an oscillation in the horizontal mode will result in no change in axis ratio should the nodal lines intersect the Cartesian axes. Given the discrete nature of the numerical simulation, it is possible that the alignment of the grid cells will favour an oscillation which results in this intersection. While any large amplitude oscillation with this alignment will be evident from inspection of the interface isosurface, small oscillations may pass unnoticed.

Several droplets ($1.75mm$, $2mm$, $3mm$) have peaks at frequencies significantly lower than expected. Closer examination reveals this deviation is likely to be due to an error in the numerical model. As described in Section 2.5 there is thought to be a problem with the model which wasn't understood until after the simulations were completed. This error results in a varying resistance to motion as the droplet interface passes through space. If a droplet has mean motion through the domain, different areas of the surface will be affected by this resistance differently. This can result in a spurious oscillation, with a frequency dependent on the speed the

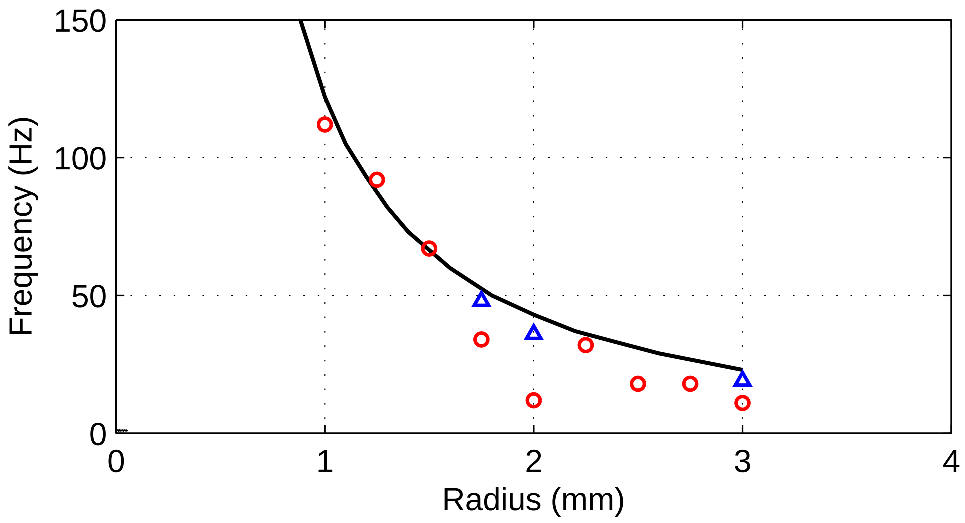

Figure 4.6.: Fundamental oscillation frequencies for 1-3mm radius droplets. In several simulations unexpected low frequency components arose and these are not thought to be physical. A potential cause of these are discussed in the text. The line represents the Rayleigh frequency, the circles represent the lowest strong oscillation frequency, and the triangles the fundamental frequency (where different from the lowest frequency).

droplet is moving through the domain, the size of the droplet, and how much resistance to motion there is. For example, in a horizontal drift the leading side of the droplet might encounter low resistance to motion, while the trailing side encounters high resistance, resulting in an increase in axis length. As the grid is uniform, it is expected that the leading side will then encounter high resistance to motion, while the trailing side encounters low resistance, resulting in a decrease in axis length. This error artificially adds a small, low frequency, signal to droplets passing through space, and it is thought that this is the cause of the unexpected peaks on the power spectral density plots.

In the three cases where this error was clearly apparent it was possible to discern peaks in the spectra nearer to the expected fundamental frequency. These are indicated by triangles in Figure 4.6. It is thought that the same error may be responsible for lower than expected frequencies in other droplets ($2.5mm$, $2.75mm$), however in these cases the error frequency and the true frequency are indistinguishable on the plots.

Other properties of the oscillations observed matched experimental data well. Figure 4.7 shows visualizations of droplets as they fall. The modes match well with both theoretical and experimental data. Figure 4.8 compares the Szakáll et al. empirical axis ratio fit to amplitudes from the simulations, showing a good match to simulation.

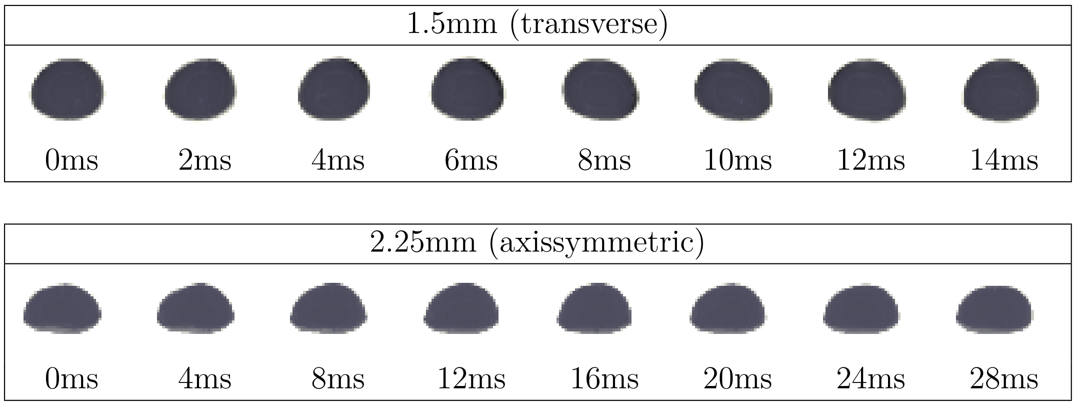

Figure 4.7.: Visualizations of droplet motion.

Although the frequencies of higher mode oscillations formed spikes in the power spectral density plots for some of the droplets simulated (1.25mm, 1.5mm and 2.25mm), no higher mode oscillations were visible by observation of the drop movements.

4.3.4. Turbulent Wake and Eddy Shedding

Theory and Experimental

As spheres fall through a medium it has been found that, for certain Reynolds numbers, an unsteady phenomenon known as eddy shedding takes place [79]. Given that freely falling water drops tend to be near-spherical, the same patterns of eddy shedding are expected to be apparent in these flows. The Strouhal number is most commonly used to non-dimensionalize the eddy shedding frequency of an object. It is defined as:

$$St = \frac{fL}{U_\infty} \tag{4.5}$$

Where L is a characteristic length scale of the flow and U_∞ the free stream velocity. In this case L is taken to be the diameter of an equivalent spherical droplet.

The Reynolds number range the simulations aim to reproduce is from $Re \approx 850$ to $Re \approx 3550$. Studies of spheres [84, 79] at these Reynolds numbers have shown there to be two modes of eddy shedding, one with a low Strouhal number (the low-mode), and one with a high Strouhal number (the high-mode). The low-mode takes the form of large scale vortices, the shedding point of which "rotates

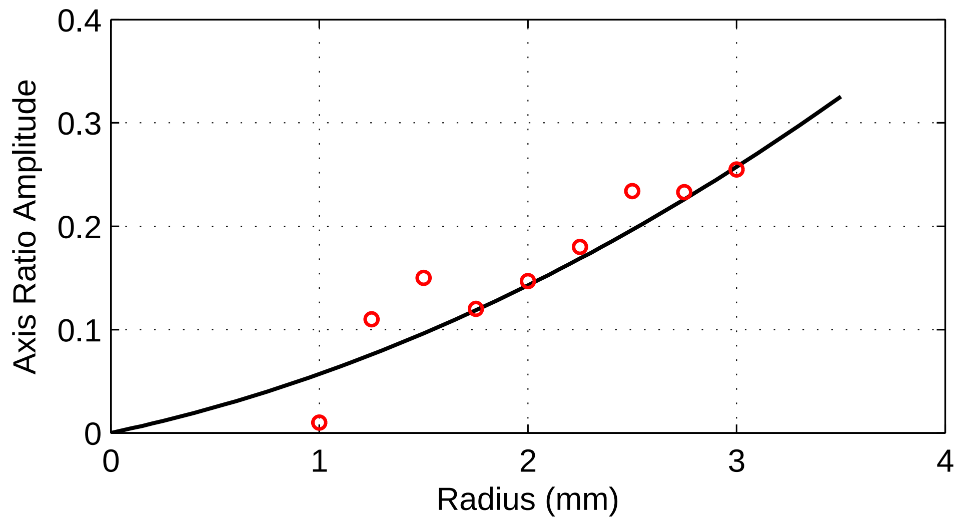

Figure 4.8.: Axis ratio peak-to-peak amplitudes for 1-3mm radius droplets (circles) compared with the polynomial fit of Szakáll et al. (line).

slowly and irregularly" around an axis though the centre of the sphere in the free-stream direction. The high-mode takes the form of smaller scale eddy shedding from the cylindrical shear-layer vortex sheet close to the separation point. It has been found that a Strouhal number for the low-mode varies almost linearly from approximately 0.2 at the lower end of Reynolds numbers to approximately 0.24 at the upper. The Strouhal number for the high mode, however, is expected to rise from 0.2 to 1 in the same range. Figure 4.9 is a reproduction of a figure presented by Sakamoto and Haniu illustrating these two shedding Strouhal numbers [79].

Although these results are important, and the assumption that a droplet behaves as a sphere seems valid for small droplets, larger droplets deviate significantly in shape from spherical and this may cause significant differences from the results for spheres.

For small, near spherical droplets (equivalent radius 0.5 to 0.8mm, $Re \approx 270$ to $Re \approx 600$), the eddy shedding frequency of droplets has been found to be similar to the eddy shedding frequency of an equivalent radius sphere [13]. There has been little research into the eddy shedding frequencies of larger droplets, with the work of Saylor and Jones being the only known reference studying vortices in larger water droplets [76]. The authors present images of vortices in the wake of a droplet of approximate Reynolds number 1000. Although the images do show clear vortices in positions not dissimilar to vortex structures one might expect from flow around a cylinder, this is perhaps misleading. As the images are stills of different droplets, it is impossible to determine how the wake was evolving over time, and therefore impossible to draw substantial conclusions.

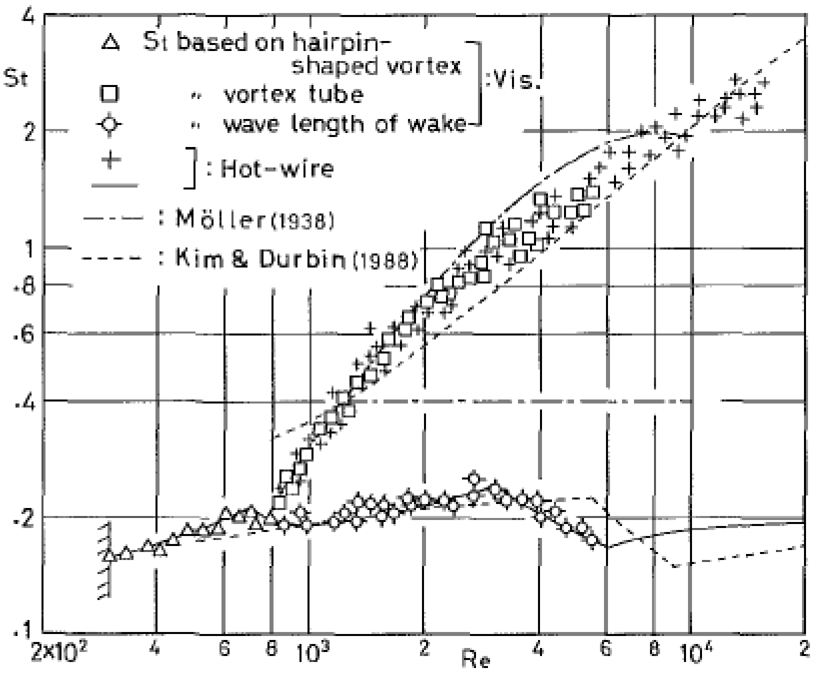

Figure 4.9.: Eddy shedding Strouhal numbers of spheres. Source: Figure 4, Sakamoto and Haniu, “A study on Vortex Shedding From Spheres in a Uniform Flow”, Page 3[79].

One significant finding of the work of Saylor and Jones was a correlation between the position of an asymmetric vortex, and the canting angle (or tilt) of the droplet, despite the expected eddy shedding frequency being several times higher than the expected resonant frequency of the droplet. While the authors suggest that the canting angle may be the result of these vortices, it may be that the causality is reversed, and that the vortices result from the canting angle.

Simulations

By calculating the power spectral density of the variation in vorticity, summed over a plane, downstream from the droplet, Strouhal numbers can be calculated for eddy shedding from droplets. Two planes were placed downstream from the centre of the droplet: the first approximately 5 radii downstream, the second approximately 10 radii downstream. The closer plane was placed with the intention of capturing high-mode oscillations before their frequency signal became dissipated. The farther plane was placed with the intention of capturing low-mode oscillations. Each plane was divided into equally sized square quarters. By summing the vorticity over each quarter individually, and calculating the frequencies at which the ratios between them change, it is possible to determine if there is any periodicity in the shedding position of the eddies.

Long term drift in droplet position caused the downstream positions of the planes relative to the droplet to vary by approximately one radius. Similar drifting resulted in the centre point of the quartered plane not being positioned above the centre of mass of the droplet. The precise position of the planes is not required for accurate sampling, and the motion itself was not relevant at the frequencies considered, although it can be seen as a low frequency term in the spectra. The power spectral densities for the nine simulated droplets are presented in Appendix C. As an example, the power spectral density of the 1.75mm radius simulated droplet is included here (Figure 4.10).

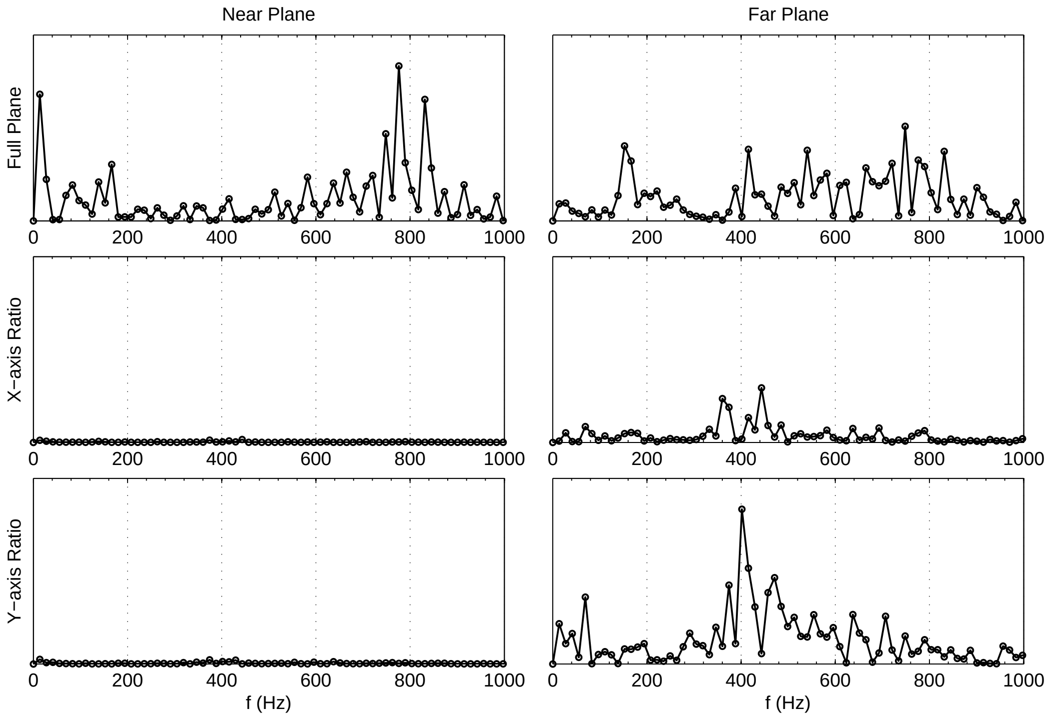

Figure 4.10.: Eddy shedding frequencies of the simulated 1.75mm radius droplet.

It was found that both Strouhal numbers expected for spherical flow were present in the wake of the droplets. Observation of the flow indicates that the upper two Strouhal numbers correspond to the expected modes for wake shedding from a sphere. One point of interest is that the transition from hair-pin shaped low Strouhal number vortex shedding to an undefined wavelength described by Sakamoto and Haniu was not as sharp as Figure 4.9 would imply, with hairpin vortices distinguishable in the flow at Reynolds numbers up to 1600. Although the domain was too small to capture more than one complete vortex loop at any given time, those single vortex loops were observed. Figure 4.11 illustrates a typical vortex loop having been shed from a 1.25mm radius droplet at a Reynolds number of approximately 1200. The pattern here is very similar to that of the formation of vortex loops observed in sphere wakes of similar Reynolds number [84, 79, 85].

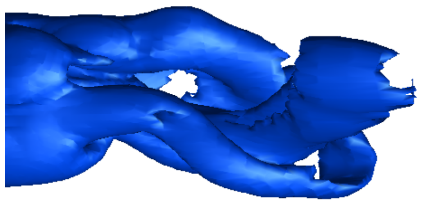

Figure 4.11.: An isosurface of vorticity downstream from a 1.25mm radius droplet. The flow direction is from left to right. Clearly visible is a vortex loop surrounding a central vortex core.

This overlap between structured eddy shedding and wake oscillation does lead to some potential confusion. While the eddies remain structured (approximately $Re < 1600$) the low-mode represents two frequencies: the frequency of eddy shedding and the frequency of oscillation of the wake. The former is double the latter, with one eddy shed for each peak of the wake oscillation. It is important to specify that the low-mode Strouhal number is defined by the frequency of oscillation of the wake (the lower of the two frequencies). Due to the two frequencies present in the low-mode, a potential problem arises when interpreting the spectra, as both frequencies will alter the ratio between the cut planes. For this reason the spectra cannot be used alone to determine the frequencies of the low-mode Strouhal number, and an observation of the flow must also be made.

As droplets oscillate, their chord length changes, and it is therefore expected that the Strouhal number also changes with time. This can be seen in Figure 4.10: it is clear from the full near plane that eddy shedding is occurring at approximately 780-840Hz, with two or three distinct peaks visible. For this reason a minimum and maximum Strouhal number has been calculated for each eddy shedding type on each droplet. These Strouhal numbers are tabulated in Table 4.2.

At high Reynolds number ($Re > 2500$) the power spectra become extremely noisy, and it becomes increasingly difficult to determine the dominant frequencies. There are two primary reasons for this. Firstly, as Reynolds number increases the wake becomes more chaotic as multiple vortex lines interact. This is expected to add noise to the frequency plot as vortices no longer pass through the planes at regular intervals, but instead experience bunching and spreading. Secondly, the oscillating axis ratio of the droplets will cause a change in frontal area with time. Larger droplets tend to oscillate more, spreading the frequencies over a larger range. For this reason it was often impossible to determine the high-mode frequency, and values for droplets of greater than 2mm radius are uncertain.

Radius (mm)	1	1.25	1.5	1.75	2	2.25	2.5	2.75	3
Re	839	1195	1595	1997	2353	2697	2990	3246	3533
$St_{1,min}$	0.16	0.14	0.13	0.16	0.16	0.17	0.17	0.15	0.14
$St_{1,max}$	0.16	0.15	0.13	0.17	0.17	0.17	0.24	0.15	0.16
$St_{2,min}$	0.21	0.24	0.20	0.31	-	0.26	0.27	0.34	-
$St_{2,max}$	0.21	0.3	0.25	0.34	-	0.33	0.28	0.37	-

Table 4.2.: Sphere-mode Strouhal numbers of the simulated droplets at terminal velocity. Many of the power spectra were unclear at high Reynolds number and hence several Strouhal numbers could not be determined.

The Strouhal numbers presented in Table 4.2 tend to be lower than those for a sphere at equivalent Reynolds number, especially for high-mode oscillations. This difference is likely to be due to droplet deformation, as droplets at higher Reynolds number have axis ratios which are quite far from spherical. This will change the properties of the wake significantly as the frontal area is increased, the separation point will move, and the recirculation region becomes a different size.

Another cause of difference between flow past a droplet and flow past a sphere are the effects of internal velocity: air passing the droplet will cause circulation within the droplet, reducing the velocity gradient normal to the interface, and therefore the shear stress. It will be shown later in Section 4.3.5 that this effect is expected to be equivalent to a decrease in free stream velocity of approximately 4%.

Despite the lower than expected values, the sphere-mode Strouhal numbers are still too high to resonate with the lower oscillation modes of the droplets. While they will still cause some motion in the droplet, it is unlikely that this will be significant.

The orientation of the wake is also of interest. In many instances it is clear that a strong asymmetric wake is present, and that, not only is the wake oscillating with a set frequency, but the oscillation has some alignment to it. Referring to Figure 4.10, a significant oscillation is visible at approximately 400Hz in the far y-axis ratio, implying an asymmetric oscillation with a path nearly parallel to the y-axis. This contrasts with experimental findings for spheres, which are described as having “irregularly” oscillating wakes [79]. It seems likely that these systematic oscillations are present due to asymmetries within the profile of the droplet favouring one direction. A correlation between the eddy shedding axis and the axis through which the main motion of droplets oscillating transversely occurs was sought, but none could be found. This suggests that while the droplet shape may favour an eddy shedding axis, this axis is not necessarily the same as (or even perpendicular to) the droplet oscillation axis. If high frequency asymmetric eddy shedding is indeed the source of transverse oscillations at a lower frequency, the mechanics which bring about this connection remain unclear.

Several of the droplets were found to have significant low frequency components in the wake. It has been observed that these low frequency components correspond to the bulk movement of the droplets within the domain, and, as described previously, these bulk movements will result in low frequency errors due to the method used to measure the frequencies. As such, it seems that these low frequencies can be discounted.

Thus far the focus of discussion has been on characterizing the two modes resulting in eddies entrained by the main flow, however close examination of the flow reveals a third mode by which eddies are produced. Figure 4.12 shows slices of the magnitude of vorticity through the X:Z plane of the 2.5mm radius simulated droplet. These plots show not only the two modes described previously, but also a third mode resulting in eddy formation within the recirculation region. These recirculation eddies are similar to those visualized by Saylor and Jones [76].

Although it is not immediately apparent, careful study of these plots reveals a frequency at which vorticity is pushed into the recirculation region. Five such events are visible in the above plot:

1. 4-8ms: Vorticity is pushed from the right shear layer into the centre of the recirculation region before dissipating.
2. 10-14ms: Vorticity is pushed from the top of the recirculation region to the right shear layer and becomes entrained.
3. 14-18ms: A similar pattern to 2.
4. 18-22ms: Vorticity is pushed from the right shear layer into the centre of the recirculation region before dissipating.
5. 22ms onward: Vorticity is pushed downward from the top of the recirculation region and dissipates slowly.

These results give an approximate frequency for eddies forming within the recirculation region of 275Hz. This value is significantly smaller than the two sphere-mode eddy shedding frequencies, and hints at a possible mechanism driving droplet oscillations. A further point of interest is the different type of events, with some vortices dissipating quickly, some becoming entrained with the shear layer, and others being much more persistent. The characterization of this mode could be important for the understanding of the driving factors behind droplet oscillation, as it appears to occur at a significantly lower frequency than the two primary oscillations. Unfortunately the time allotted for this project does not allow for any further study to be made here.

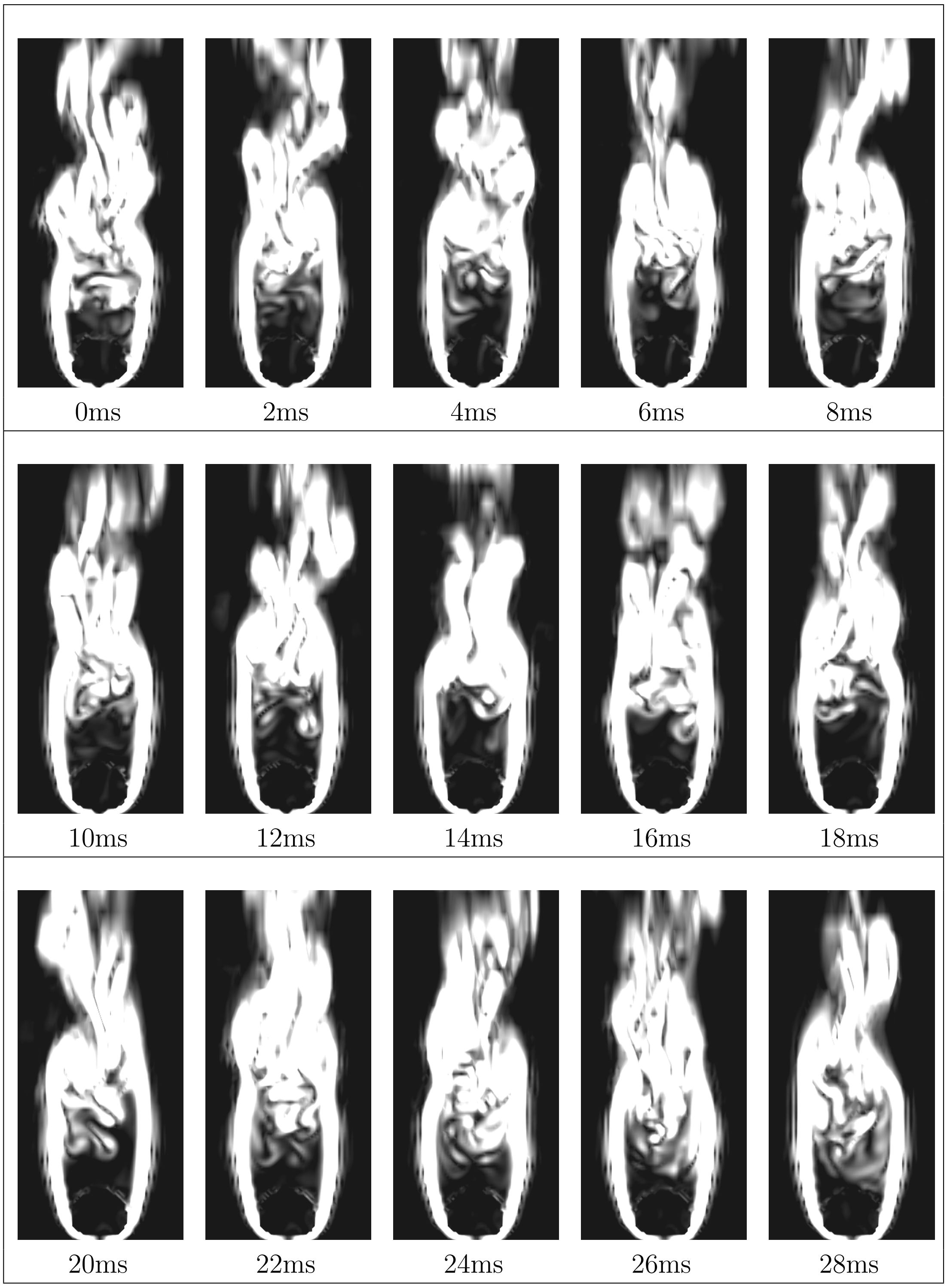

Figure 4.12.: Vorticity shedding and recirculation within the wake of the 1.75mm droplet. Gravity is acting downward, and the droplet is visible as a darkened area at the bottom of each plot. The scale is set to allow small vortices to be visible, with fully white regions representing a vorticity of greater than $5000s^{-1}$. With an interval of 2ms between plots, the high-mode is not easily discernible, though the low-mode can be seen in the oscillation of the wake.

4.3.5. Internal Circulation

Theory and Experimental

The internal circulation of water within a falling droplet has not been studied in as much depth as other properties, and its interaction with the droplet's other properties is still unclear. Although several authors have studied internal circulation of water droplets [60, 86, 66], their studies have largely focused on droplets below the size range simulated here.

The work of LeClair et al. presents experimental data and a theoretical model for the tangential surface velocity within the droplet [86]. Above a Reynolds number of 400, it is expected that the mean tangential velocity at any given azimuth is simply proportional to the velocity of the droplet, with a peak non-dimensional tangential velocity ($^{u_t}/_{U_\infty}$) of approximately 0.043$^{\mathrm{m}}/_{\mathrm{s}}$. Szakáll et al. showed, by comparing this theory with experimental data, that a fairly good match was made between the theoretical maximum tangential velocity and the experimental maximum [66], although their experimental maximums fell short of the values predicted by LeClair et al..

Szakáll et al. also allude to a potential oscillation in the internal circulation, whereby a coupling occurs between the primary circulation within the droplet and a second opposing circulation at the rear. It is reported that these circulations have a cyclic behaviour, with the rearward circulation gaining strength, coupling with the primary circulation, breaking down into chaotic flow, decaying, and then leading back to the formation of primary and secondary circulations.

Simulations

The maximum internal speed in the droplet size range simulated is given in Figure 4.13. The results were calculated as the mean maximum value over a period of approximately 250ms. Good agreement is obtained between simulated and theoretical results, though theoretical results appear to be a lower bound. This may be due to the oscillatory behaviour of droplets: the LeClair et al. model does not account for large movements of the interface, and these are likely to increase internal tangential velocity relative to steady results.

The cyclic behaviour reported by Szakáll et al. could not be detected. Figure 4.14 illustrates three stills of the 1.75mm simulated droplet. Nothing approaching the primary/secondary structure described by Szakáll et al. could be found, though they do not specify the maximum Reynolds number at which they occur, only that the minimum is 300. It could be that a maximum Reynolds number for the cyclic behaviour exists, and that all the simulations exceeded that Reynolds number. The pattern was similar in every droplet for which internal circulation

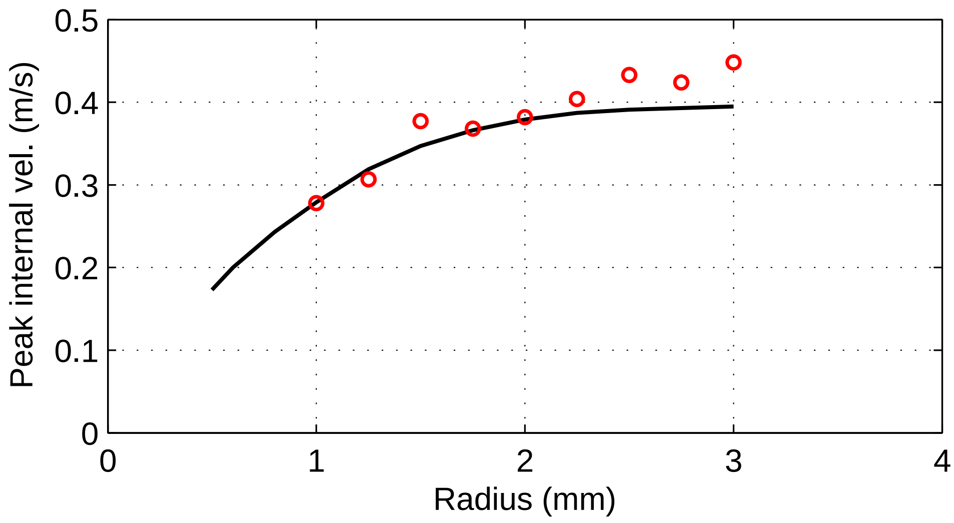

Figure 4.13.: Maximum internal tangential velocity. The circles represent simulated data, the solid line the theoretical values from LeClair et al.[86]

was visualized: flow resembling the external flow on the upwind and side sections, with a vortex ring centred near the band of peak horizontal chord. The flow at the rear was unstable, with occasional vortex structures internally. No pattern could be found in the internal vortex structures, though it is possible that one exists.

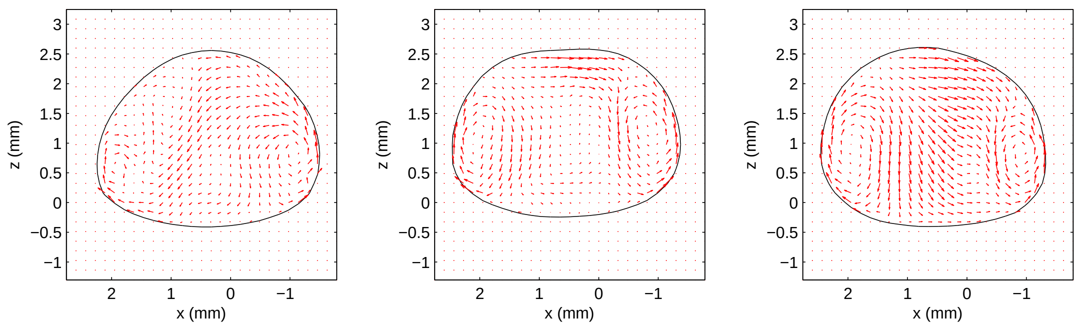

Figure 4.14.: Velocity field within the 1.75mm radius droplet at three different time steps. The slice is taken through the centre of mass of the droplet. The time interval between each is 50ms.

How internal circulation affects external flow patterns, such as the wake behaviour and droplet oscillations, is unclear and requires further analysis. It is not expected that the effects will be large, as the peak internal velocities are small when compared to the external flows. For example, the internal velocity acts to reduce the shear stress across the interface, lowering the vorticity imparted to the air as it passes. This will have an effect on eddy shedding frequencies, however as the peak internal velocity is only approximately 4% of the free-stream velocity, the effect is expected to be small.

4.4. Summary

Water droplets with spherical radii of $1mm$ to $3mm$ were simulated under free-fall conditions in air. The Reynolds number of the system varied from $Re \approx 850$ to $Re \approx 3500$. The simulations were designed to mimic natural raindrops of different sizes falling at sea level. Various properties of the system were examined in detail, and, where possible, compared to experimental results. It was found that, in most aspects, the numerical model either accurately reproduced the results of experiments or brought potentially new insight. A summary of successful results from each subsection follows:

- The terminal velocities of the droplets was found to match the experimental results of Gunn and Kinzer [58] to within 3%.
- The Beard and Chuang model [12] was found to be a good predictor of the mean droplet axis ratio and shape.
- Fundamental droplet oscillations were found to match with theoretical and experimental results. The amplitude of the oscillations was found to correspond with an empirical curve derived from experimental data. Indications of higher mode oscillations were seen on a few droplets.
- Eddy shedding modes corresponding to the eddy shedding modes of spheres at similar Reynolds numbers were seen. Although the Strouhal numbers of these modes did not match experimental results for spheres, this is not unexpected as the system is different in several ways. Not only do droplets flatten, altering frontal area, separation points and the size of the recirculation region, but droplets also have internal circulation which acts to decrease the shear stress on the air as it flows past. The Strouhal numbers of droplet eddy shedding were quantified, and orientation was found in the wake. Eddy formation within the recirculation region was also briefly examined, and it is thought that a mechanism might exist in this region resulting in forcing of droplet oscillations.
- The internal circulation speed was found to agree with the results of LeClair et al. [86], especially for smaller drop radius. Although no quantitative conclusions were made regarding the influence of internal circulation on either droplet shapes, oscillations or external flow, the model has proved that it is capable of producing data to analyze this influence should a hypothesis be made.

Although the simulations reproduced most expected results, there were some inconsistencies:

- The numerical model failed to predict the large horizontal drift seen in some experimental results. One potential explanation for this difference is the method by which the simulations were initialized. By initializing the droplets at a significant proportion of their terminal velocity, a low Reynolds number region where asymmetric vortex shedding occurs is not simulated. As asymmetric vortices will tend to lead to horizontal drift it may be that the simulations unintentionally circumvent the cause of the drift.
- The oscillation frequencies were found to match theory, however the quality of the results was poor. Several spectra contain unexpected oscillations, which in some cases overwhelmed the expected frequencies. It is thought that the cause of this error is the "straightening" effect, that was observed when studying the rise of bubbles in viscous fluids (Section A.2), and discussed in Section 2.5. Although no similar effect was observed when visualizing the simulated droplets, the amplitude of the natural oscillations is expected to be of the order of only a few grid cells. Any unnatural resistance to movement of the interface at this level, even if the influence is slight, could cause significant errors in the calculated oscillation frequencies. If the numerical error described is indeed the cause of these spurious oscillations, any future work must first correct this error before studying oscillations as, while many of the droplets performed as expected while oscillating, many did not.
- No oscillation could be found in the internal circulation of the droplets. Although no experimental data specifically reports the existence of a periodic oscillation in internal circulation at the Reynolds numbers that have been simulated, an oscillation has been reported to exist at lower Reynolds number. No upper bound was given for the Reynolds number of this oscillation, and it may be that one exists [66]. One way to determine whether the numerical simulation is accurate in this regard would be to simulate smaller radius droplets and observe their behaviour. If smaller droplets were found to exhibit an oscillation, it could be said with more confidence that the numerical model was predicting physical internal circulation at the larger droplet sizes, and an upper bound could be found of the Reynolds number required for cyclic behaviour.

One of the main advantages of CFD over experimental work is the amount of information one can collect without causing any disturbance to the flow. This allowed for in-depth analysis of the entire system, and specifically of the wake of the droplets, in a manner which is significantly harder to achieve experimentally. The observation of a third mode of eddy generation is important, as this mode may be a significant contributor to driving droplet oscillation. While it was not studied in depth here, it is hoped that any authors who continue this work are able to study it in more depth than was possible here.

Many of the results found in this work would be improved if the simulations had either been run for a longer, or been run with faster hardware. One of decisions made before collecting results was to spread the computational resources available over a spectrum of droplets. While this has allowed a greater number of scenarios to be studied, it inevitably reduces the amount and accuracy of data that can be gathered from any single simulation. These studies have shown that the numerical model is capable of accurately simulating freely falling water droplets, with the simulations matching experimental and theoretical results over a wide range of droplet sizes.

5. Conclusions and Future Work

5.1. Conclusions

The aims of this work were threefold: to develop numerical methods capable of simulating three-dimensional immiscible flows, to implement these methods in an efficient manner on a GPU based system, and to use these methods to model a problem which has not yet been modeled using CFD. All three aims have been achieved, requiring novel methods and discovering new information about the test problem.

Numerical Methods

The numerical methods used required a combination of several complex methods. An incompressible pressure-projection method designed for curvature driven flows was combined with a conservative level set method, with the aim of accurately simulating subtle interactions at an interface with very severe jumps in properties. Not only is such a combination novel, but there are novel additions to several parts of the scheme. Neither the curvature nor viscosity interpolation method described have been seen in the literature, nor has the conservative level set cell mixing procedure previously been applied to the pressure projection method.

The resulting numerical scheme was shown to be capable of accurately simulating both inviscid surface tension driven flows (a zero-gravity oscillating water drop, Section A.1) and highly viscous low surface tension flows (air bubbles rising in viscous fluids Section A.2). In both domains it was found to perform well, predicting a frequency very close to the theoretical natural frequency for the oscillating water drop, and accurately predicting both the terminal velocities and shapes of the bubbles. Slight errors were found in the shape of the bubbles, with unphysical "straightening" of the interface. This is thought to result from numerical jumps caused by the conservative level set method. As the level set field moves through space, cells close to the interface undergo a merging procedure so as to conserve momentum. Unfortunately this procedure results in a sharp jump in local velocity at the moment of merging. This jump is thought to cause small oscillations of the interface position about the merging point, which can unphysically constrain the position of the interface.

Although this error was not visibly apparent in the main test case, it is thought that it may be a cause of the relatively poor results for droplet oscillations when placed in a flow.

GPU Implementation

Although initial work suggested that the GPU performance was likely to be better than that which was achieved, the GPUs were still able to outperform a single core of a CPU by an order of magnitude. The reasons behind the unexpectedly poor performance are twofold. Firstly, the complete code was significantly more complicated than the test case, and these complexities resulted in unexpected performance problems. Secondly, given the extremely rapid development in technology it is very hard to find a fair point of comparison between CPUs and GPUs. As new hardware is released the performance balance shifts between the competing hardware types and, as such, a fair comparison is problematic. It is very possible that the initial tests were “unfair” in favour of the GPUs, whereas the current results are “unfair” in favour of the CPUs.

Despite the slight disappointment in performance the order of magnitude increase over a single core was still a of significant benefit. This acceleration allowed for single-node computation of the falling droplet problem, avoiding complications due to low bandwidth and high latency inter-node connections. Instead, a multi-realization approach was taken, with each of the eight GPUs available simulating a different set of initial conditions simultaneously.

This multi-realization approach increased efficiency as several of the methods used are not very suitable for node-level parallelism. This is either because they require frequent transfers of information, or because they are only necessary in a certain part of the domain, typically near the interface.

Falling Droplets

Water droplets with radii in the range $1mm$ to $3mm$ were simulated in free fall conditions. Comparison with experimental results clearly shows that the numerical methods developed were indeed capable of solving a complex interfacial problem, with agreement in nearly every aspect.

The wake of the droplets was examined in depth and it is thought that this work will bring about a significant advancement in understanding in this area. It has been found that, although the droplet wakes are similar to those of spheres, the frequencies present in the wake are significantly lower than reported eddy shedding frequencies of spheres. The wakes were also found to have an oriented asymmetric structure, with large scale oscillations along a line on the X-Y plane.

Eddy formation within the recirculation region was briefly examined, and a low frequency oscillation described. It is thought that this oscillation may be a factor driving droplet oscillations, however there was insufficient time to study it in depth.

5.2. Future Work

Numerical Methods

The most apparent issue with the numerical methods is the treatment of viscosity: all other proprieties are treated sharply at the interface, but viscosity is treated as a continuously varying property. While it was found that viscous problems could still be modeled fairly accurately (Section A.2) there were some clear errors apparent. A new method would be required to model interface viscosity, as current sharp viscosity methods are not compatible with the conservative level set method.

Another significant area of concern regarding the numerical methods is the numerical jump introduced by the conservative level set method. This problem is thought to be the reason why droplet oscillations are incorrectly simulated, so it is important to find a solution.

GPU Implementation

While the GPU implementation of the numerical methods was found to give an order of magnitude speedup over a single core, this can undoubtedly be improved. It was found that, on both CPU and GPU, the pressure projection step took a significantly larger proportion of runtime than other steps, despite the core of the method, the BiCGStab solver performing very well on the GPU. This suggests that there may be inefficiencies elsewhere in the method that could be eliminated.

Another area which could be examined is whether any features of the newer NVIDIA Kepler GPUs[87] could be used to accelerate any of the methods.

Falling Droplets

As the work presented here is the first set of CFD simulations studying full flow around a three dimensional droplet, there are many extensions that are possible. Perhaps the most obvious is the detail at which an individual droplet is studied. The work presented here concentrated on the simulation of a range of droplet sizes. This meant that the processing power available to a single droplet was only a fraction of the processing power available. A simple extension to this work would be to change the scope, and instead simulate one droplet in greater detail: either at a higher resolution or for a longer period of time. If the droplets are to be studied for a longer period of time one potential issue is the poor performance of GPUs for small grid sizes. By increasing the number of GPUs working on a single problem, the efficiency of each GPU is reduced. It may be that additional work would be required to accelerate the GPU methods in multi-node environments.

One area that may be of particular interest is further study into recirculation eddies. Early indications are that a low frequency mode exists within the recirculation region which may be a significant factor in driving droplet oscillations. Unfortunately the time available for this work did not allow for this mode to be studied in as much detail as was desired. Further analysis would require collection of higher frequency data than has been collected thus far.

There is active research within droplet dynamics on the breakup of larger droplets into smaller droplets. While the work contained within this thesis was focused on stable droplets, it has been reported that an increase in radius beyond the maximum radius studied here will shortly lead to droplet breakup [58, 88]. The modes of breakup and the drop size distribution in the remaining spray are both areas of interest and the problem is well suited to the computational model presented in this thesis. While it is likely that a few changes would be required to account for some of the properties of such systems it is anticipated that these changes need not be significant. Specifically, it is likely that the mass correction method would need to be updated to take into account multiple convex interfaces, and that the cell merging method would have to be made more robust, as currently it is unclear what the procedure should be if there are no neighbouring cells to merge a small cell with.

It may be of interest to study droplets in a more realistic environment. The impacts of interaction between multiple droplets, or of atmospheric effects could significantly alter the behaviour of the system. It is expected that the addition of either factor would add significant forcing to the droplets, likely causing an increase in droplet oscillation, and potentially breakup. Finally, one could explore of the parameter space of the system. While for larger droplets it is expected that results will be similar for similar Eötvös numbers, there remain plenty of areas where property changes could bring about interesting results. For example it may be interesting to simulate lower surface tension fluids, such as ethanol.

References

[1] D. Bhaga and M. E. Weber, “Bubbles in viscous liquids: shapes, wakes and velocities,” *Journal of Fluid Mechanics*, vol. 105, pp. 61–85, 1981.

[2] H. E. White, *Modern College Physics.* Van Nostrand, (1948).

[3] Young, *Phil. Trans*, p. 65, 1805.

[4] Laplace, *Mécanique céleste*, 1805.

[5] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, “Real-time robot motion planning using rasterizing computer graphics hardware,” in *In Proc. SIGGRAPH*, 1990, pp. 327–335.

[6] H.-H. Hsieh and W.-K. Tai, “A simple gpu-based approach for 3D voronoi diagram construction and visualization,” *Simulation Modelling Practice and Theory*, vol. 13, no. 8, pp. 681 – 692, 2005.

[7] M. Schatz, C. Trapnell, A. Delcher, and A. Varshney, “High-throughput sequence alignment using graphics processing units,” *BMC Bioinformatics*, vol. 8, no. 1, p. 474, 2007.

[8] R. V. van Nieuwpoort and J. W. Romein, “Using many-core hardware to correlate radio astronomy signals,” in *Proceedings of the 23rd international conference on Supercomputing*, ser. ICS ’09. New York, NY, USA: ACM, 2009, pp. 440–449.

[9] T. Preis, P. Virnau, W. Paul, and J. J. Schneider, “Gpu accelerated monte carlo simulation of the 2D and 3D ising model,” *J. Comput. Phys.*, vol. 228, no. 12, pp. 4468–4477, Jul. 2009.

[10] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with CUDA,” *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[11] “NVIDIA GTC 2012 Keynote,” Presentation, May 2012.

[12] K. V. Beard and C. Chuang, “A new model for the equilibrium shape of raindrops,” *Journal of the Atmospheric sciences*, vol. 44, no. 11, pp. 1509–1524, 1987.

[13] K. V. Beard, V. Bringi, and M. Thurai, “A new understanding of raindrop shape,” *Atmospheric Research*, vol. 97, no. 4, pp. 396–415, 2010.

[14] S. Osher and J. A. Sethian, “Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations,” *Journal of Computational Physics*, vol. 79, no. 1, pp. 12 – 49, 1988.

[15] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, “A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method),” *Journal of Computational Physics*, vol. 152, no. 2, pp. 457–492, 1999.

[16] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, “A hybrid particle level set method for improved interface capturing,” *Journal of Computational Physics*, vol. 183, no. 1, pp. 83–116, 2002.

[17] E. Bertakis, S. Groß, J. Grande, O. Fortmeier, A. Reusken, and A. Pfennig, “Validated simulation of droplet sedimentation with finite-element and level-set methods,” *Chemical Engineering Science*, vol. 65, no. 6, pp. 2037–2051, 2010.

[18] R. Nourgaliev and T. Theofanous, “High-fidelity interface tracking in compressible flows: Unlimited anchored adaptive level set,” *Journal of Computational Physics*, vol. 224, no. 2, pp. 836 – 866, 2007.

[19] P. Macklin and J. Lowengrub, “An improved geometry-aware curvature discretization for level set methods: application to tumor growth,” *journal of Computational Physics*, vol. 215, no. 2, pp. 392–401, 2006.

[20] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J.-F. Remacle, “A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics,” *Journal of Computational Physics*, vol. 225, no. 1, pp. 949 – 974, 2007.

[21] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces.* Springer, 2003, vol. 153.

[22] X. Hu, B. Khoo, N. Adams, and F. Huang, “A conservative interface method for compressible flows,” *Journal of Computational Physics*, vol. 219, no. 2, pp. 553–578, 2006.

[23] P. Barton, B. Obadia, and D. Drikakis, “A conservative level-set based method for compressible solid/fluid problems on fixed grids,” *Journal of Computational Physics*, vol. 230, no. 21, pp. 7867–7890, 2011.

[24] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

[25] D. Hartmann, M. Meinke, and W. Schröder, “The constrained reinitialization equation for level set methods,” *Journal of computational physics*, vol. 229, no. 5, pp. 1514–1535, 2010.

[26] C. Walker and B. Müller, "Constrained reinitialisation of the conservative level set method," in *Proceedings of the 8th International Conference on CFD in Oil & Gas, Metallurgical and Process Industries*, 2011.

[27] S. K. Godunov, "A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics," *Matematicheskii Sbornik*, vol. 47, pp. 357–393, 1959.

[28] K. H. Kim and C. Kim, "Accurate, efficient and monotonic numerical methods for multi-dimensional compressible flows: Part II: Multi-dimensional limiting process," *Journal of Computational Physics*, vol. 208, no. 2, pp. 570–615, 2005.

[29] V. V. Rusanov, *Calculation of interaction of non-steady shock waves with obstacles.* NRC, Division of Mechanical Engineering, 1962.

[30] D. Drikakis and W. Rider, *High-resolution methods for incompressible and low-speed flows.* Springer, 2005.

[31] M. Kang, R. P. Fedkiw, and X.-D. Liu, "A boundary condition capturing method for multiphase incompressible flow," *Journal of Scientific Computing*, vol. 15, no. 3, pp. 323–360, 2000.

[32] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, no. 2, pp. 439–471, 1988.

[33] A. J. Chorin, "Numerical solution of incompressible flow problems," *Studies in Numerical Analysis*, vol. 2, pp. 64–71, 1968.

[34] ——, "On the convergence of discrete approximations to the navier-stokes equations," *Mathematics of Computation*, vol. 23, no. 106, pp. 341–353, 1969.

[35] J. B. Bell, P. Colella, and H. M. Glaz, "A second-order projection method for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 85, no. 2, pp. 257–283, 1989.

[36] J. B. Bell and D. L. Marcus, "A second-order projection method for variable-density flows," *Journal of Computational Physics*, vol. 101, no. 2, pp. 334–348, 1992.

[37] C.-W. Shu *et al.*, "A numerical resolution study of high order essentially non-oscillatory schemes applied to incompressible flow," *Journal of Computational Physics*, vol. 110, no. 1, pp. 39–46, 1994.

[38] M. Andrews, "Accurate computation of convective transport in transient two-phase flow," *International journal for numerical methods in fluids*, vol. 21, no. 3, pp. 205–222, 1995.

[39] Y. Saad and Y. Saad, *Iterative methods for sparse linear systems.* PWS publishing company Boston, 1996, vol. 620.

[40] X.-D. Liu, R. P. Fedkiw, and M. Kang, “A boundary condition capturing method for poisson’s equation on irregular domains,” *Journal of Computational Physics*, vol. 160, no. 1, pp. 151 – 178, 2000.

[41] H. A. Van der Vorst, “Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,” *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, 1992.

[42] T. Brandvik and G. Pullan, “Acceleration of a two-dimensional euler flow solver using commodity graphics hardware,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 221, no. 12, pp. 1745–1748, 2007.

[43] E. Elsen, P. LeGresley, and E. Darve, “Large calculation of the flow over a hypersonic vehicle using a GPU,” *Journal of Computational Physics*, vol. 227, no. 24, pp. 10 148–10 161, 2008.

[44] J. Cohen and M. J. Molemaker, “A fast double precision CFD code using CUDA,” *Parallel Computational Fluid Dynamics: Recent Advances and Future Directions*, pp. 414–429, 2009.

[45] Portland group accelerator. [Online]. Available: http://www.pgroup.com/resources/accel.htm

[46] Astral supercomputer. [Online]. Available: http://www.top500.org/system/175535

[47] T. Brandvik and G. Pullan, “An accelerated 3D navier-stokes solver for flows in turbomachines,” *ASME Conference Proceedings*, vol. 2009, no. 48883, pp. 619–629, 2009.

[48] P. Micikevicius, “3D finite difference computation on GPUs using CUDA,” in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, ser. GPGPU-2. New York, NY, USA: ACM, 2009, pp. 79–84.

[49] J. D. McCalpin, “Memory bandwidth and machine balance in current high performance computers,” *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, Dec. 1995.

[50] J. Dongarra, “Preface: Basic linear algebra subprograms technical (blast) forum standard,” *International Journal of High Performance Computing Applications*, vol. 16, no. 2, pp. 115–115, 2002.

[51] cuBLAS. [Online]. Available: http://developer.nvidia.com/cublas

[52] L. Thomas, “Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory,” *Columbia Univ., NY*, 1949.

[53] J. Appleyard, J. Appleyard, M. Wakefield, and A. Desitter, “Accelerating reservoir simulators using GPU technology,” in *SPE Reservoir Simulation Symposium*, 2011.

[54] D. Atlas and C. W. Ulbrich, *Early foundations of the measurement of rainfall by radar*, 1990.

[55] P. Lenard, *Über regen*, 1904.

[56] W. Schmidt, "Eine unmittelbare bestimmung der fallgeschwindigkeit von regentropfen," *Meteorologische Zeitschrift*, vol. 26, pp. 183–184, 1909.

[57] J. O. Laws, "Measurements of the fall-velocity of water-drops and raindrops," *Transactions, American Geophysical Union*, vol. 22, pp. 709–721, 1941.

[58] R. Gunn and G. D. Kinzer, "The terminal velocity of fall for water droplets in stagnant air." *Journal of Atmospheric Sciences*, vol. 6, pp. 243–248, 1949.

[59] D. C. Blanchard, "The behavior of water drops at terminal velocity in air," *Transactions, American Geophysical Union*, vol. 31, pp. 836–842, 1950.

[60] H. Pruppacher and K. Beard, "A wind tunnel investigation of the internal circulation and shape of water drops falling at terminal velocity in air," *Quarterly Journal of the Royal Meteorological Society*, vol. 96, no. 408, pp. 247–256, 1970.

[61] H. Pruppacher and R. Pitter, "A semi-empirical determination of the shape of cloud and rain drops," *Journal of the atmospheric sciences*, vol. 28, no. 1, pp. 86–94, 1971.

[62] J. Q. Feng and K. V. Beard, "A perturbation model of raindrop oscillation characteristics with aerodynamic effects," *Journal of the atmospheric sciences*, vol. 48, no. 16, pp. 1856–1868, 1991.

[63] K. V. Beard, R. J. Kubesh, and H. T. Ochs III, "Laboratory measurements of small raindrop distortion. part i: Axis ratios and fall behavior." *Journal of Atmospheric Sciences*, vol. 48, pp. 698–710, 1991.

[64] K. V. Beard and R. J. Kubesh, "Laboratory measurements of small raindrop distortion. part 2: Oscillation frequencies and modes," *Journal of the atmospheric sciences*, vol. 48, no. 20, pp. 2245–2264, 1991.

[65] K. Andsager, K. V. Beard, and N. F. Laird, "Laboratory measurements of axis ratios for large raindrops," *J. Atmos. Sci.*, vol. 56, no. 15, pp. 2673–2683, Aug. 1999.

[66] M. Szakáll, K. Diehl, S. K. Mitra, and S. Borrmann, "A wind tunnel study on the shape, oscillation, and internal circulation of large raindrops with sizes between 2.5 and 7.5 mm," *Journal of the Atmospheric Sciences*, vol. 66, no. 3, pp. 755–765, 2009.

[67] A. Tokay and K. V. Beard, "A field study of raindrop oscillations. part I: Observation of size spectra and evaluation of oscillation causes," *Journal of applied meteorology*, vol. 35, no. 10, pp. 1671–1687, 1996.

[68] M. Thurai and V. Bringi, “Drop axis ratios from a 2D video disdrometer,” *Journal of Atmospheric and Oceanic Technology*, vol. 22, no. 7, pp. 966–978, 2005.

[69] F. Testik, A. Barros, and L. Bliven, “Field observations of multimode raindrop oscillations by high-speed imaging,” *Journal of the atmospheric sciences*, vol. 63, no. 10, pp. 2663–2668, 2006.

[70] G.-J. Huang, V. Bringi, and M. Thurai, “Orientation angle distributions of drops after an 80-m fall using a 2d video disdrometer,” *Journal of Atmospheric and Oceanic Technology*, vol. 25, no. 9, pp. 1717–1723, 2008.

[71] M. Szakáll, S. K. Mitra, K. Diehl, and S. Borrmann, “Shapes and oscillations of falling raindrops - a review,” *Atmospheric Research*, vol. 97, no. 4, pp. 416 – 425, 2010.

[72] B. Helenbrook and C. Edwards, “Quasi-steady deformation and drag of uncontaminated liquid drops,” *International journal of multiphase flow*, vol. 28, no. 10, pp. 1631–1657, 2002.

[73] J. Q. Feng, “A deformable liquid drop falling through a quiescent gas at terminal velocity,” *Journal of Fluid Mechanics*, vol. 658, pp. 438–462, 2010.

[74] J. Q. Feng and K. V. Beard, “Raindrop shape determined by computing steady axisymmetric solutions for navier–stokes equations,” *Atmospheric Research*, vol. 101, no. 1, pp. 480–491, 2011.

[75] D. J. Lycett-Brown and K. Luo, “Three-dimensional micro-droplet collision simulation using the lattice boltzmann method,” in *3rd Micro and Nano Flows Conference, Thessaloniki, Greece, 22-24 August 2011*, 2011.

[76] J. Saylor and B. Jones, “The existence of vortices in the wakes of simulated raindrops,” *Physics of Fluids*, vol. 17, p. 031706, 2005.

[77] R. Clip, J. Grace, and M. Weber, “Bubbles, drops, and particles,” 1978.

[78] R. Gunn, “Mechanical resonance in freely falling raindrops,” *Journal of Geophysical Research*, vol. 54, no. 4, pp. 383–385, 1949.

[79] H. Sakamoto and H. Haniu, “A study on vortex shedding from spheres in a uniform flow,” *ASME, Transactions, Journal of Fluids Engineering*, vol. 112, pp. 386–392, 1990.

[80] L. Rayleigh, “Xx. on the equilibrium of liquid conducting masses charged with electricity,” *Philosophical Magazine Series 5*, vol. 14, no. 87, pp. 184–186, 1882.

[81] R. J. Kubesh and K. V. Beard, “Laboratory measurements of spontaneous oscillations for moderate-size raindrops,” *Journal of the atmospheric sciences*, vol. 50, no. 8, pp. 1089–1098, 1993.

[82] K. V. Beard, “Raindrop oscillations: Evaluation of a potential flow model with gravity.” *Journal of Atmospheric Sciences*, vol. 41, pp. 1765–1774, 1984.

[83] M. Thurai, V. Bringi, M. Szakáll, S. Mitra, K. Beard, and S. Borrmann, “Drop shapes and axis ratio distributions: Comparison between 2D video disdrometer and wind-tunnel measurements,” *Journal of Atmospheric and Oceanic Technology*, vol. 26, no. 7, pp. 1427–1432, 2009.

[84] E. Achenbach, “Vortex shedding from spheres,” *Journal of Fluid Mechanics*, vol. 62, no. 02, pp. 209–221, 1974.

[85] R. Mittal and F. Najjar, “Vortex dynamics in the sphere wake,” *Red*, vol. 2, p. 2, 1999.

[86] B. LeClair, A. Hamielec, H. Pruppacher, and W. Hall, “A theoretical and experimental study of the internal circulation in water drops falling at terminal velocity in air.” *Journal of Atmospheric Sciences*, vol. 29, pp. 728–740, 1972.

[87] NVIDIA, “NVIDIA’s next generation CUDA compute architecture: Kepler GK110,” 2013.

[88] E. Villermaux and B. Bossa, “Single-drop fragmentation determines size distribution of raindrops,” *Nature Physics*, vol. 5, no. 9, pp. 697–702, 2009.

A. Validation

A.1. Surface Tension

Surface tension was validated by simulating the oscillation of a drop of fluid in a zero-gravity field. The natural frequencies a given drop of radius r is expected to be close to those given by Rayleigh[80]:

$$\omega_0 = \sqrt{\frac{\sigma i(i-1)(i+2)}{\rho r^3}} \tag{A.1}$$

The first non-trivial natural frequency is found by substituting $n = 2$:

$$\omega_{0,2} = \sqrt{\frac{8\sigma}{\rho r^3}} \tag{A.2}$$

An ellipsoidal water droplet with two semi-principle axes of length 0.95mm with the third of length 1.1mm was initialized at rest. The density of the simulated droplet was 1000 kg/m^{3}and its surface tension 0.07564 $^{\mathrm{N}}\!/_{\mathrm{m}}$. The viscosity of the droplet was negligible. The domain was discretized with a uniform grid with 64 points spanning a side length of 2.5mm in each dimension.

The system was advanced in time and the lengths of the semi-principle axes was measured. The axis ratio is then calculated by dividing the length of the original semi-major axis by the mean length of the other two axes. Substituting the properties of the droplet in to Equation A.2 this axis ratio is expected to oscillate at a frequency of approximately 780 rads^{-1}, or 124 Hz. This corresponds to a full oscillation in a period of 8.05ms. Figure A.1 shows the variation in axis ratio with time over 50ms, approximately six full oscillations.

The measured frequency from this plot is approximately 121 Hz, an error of under 3%. It is of note that the oscillation appears undamped, suggesting that the numerical scheme has very low numerical viscosity.

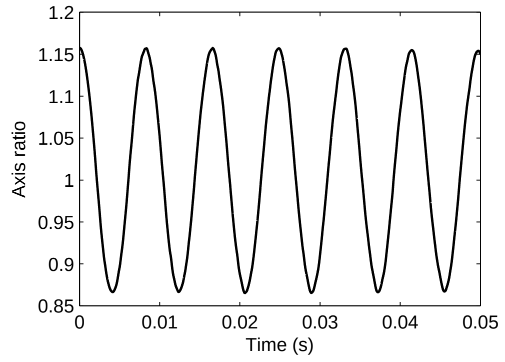

Figure A.1.: Axis ratio of droplet at rest over six oscillations.

A.2. Rising Bubble

The second validation case simulated was the motion of an air bubble rising through a viscous fluid. The objective is to reproduce some of the experimental results presented by Bhaga and Webber [1].

The results of two simulations are presented. Both are the solutions after a spherical bubble initialized at rest was allowed to rise under gravity until the interface position reached steady-state. As with the falling droplet case examined in Chapter 4, the domain is rectiliniear with spatially constant velocity boundary conditions enforced on each face.

The parameters of the problem are defined by three non-dimensional properties: the Reynolds number (Equation A.3), the Eötvös number (Equation A.4) and the Morton number (Equation A.5):

$$Re = \frac{\rho d_0 U}{\mu} \tag{A.3}$$

$$E\ddot{o} = \frac{g d_0^2 \Delta\rho}{\sigma} \tag{A.4}$$

$$Mo = \frac{g\mu^4}{\rho\sigma^3} \tag{A.5}$$

The values of these properties for the two experiments are tabulated in Table A.1. It should be noted that the Reynolds number is not explicitly set as the velocity of the bubble is a variable which is calculated based on the other parameters.

	Eö	Mo	Re
Case 1	116	266	3.57
Case 2	116	5.51	13.3

Table A.1.: Non-dimensional properties of the two test cases.

Although the falling droplet test case has significantly different non-dimensional numbers, the rising bubble test case is helpful for two main reasons. Firstly, because the Reynolds number is much lower, it acts as a validation of the viscosity model. The falling droplet test case tends to have Reynolds numbers of one to three orders of magnitude higher, and if the viscosity model is sufficient for a low Reynolds number case, it follows it should be sufficient for the falling droplet case. Secondly, the surface tension term in this test case is a lot less significant, due to the significantly higher Eötvös number. This means that the time step restriction for stability is much less strict and these simulations can be run in a much shorter time than the falling droplet case.

The first step is to validate the Reynolds number calculated from simulation. Table A.2 compares the expected Reynolds numbers with the Reynolds numbers calculated at terminal velocity from the simulation. Although the Reynolds numbers differ slightly the error is small. The second point of comparison is the bubble profiles. Figure A.2 and Figure A.3 compare the calculated interface position to photographs taken by Bhaga and Webber for case 1 and case 2 respectively.

	Expected Re	Simulated Re
Case 1	3.57	3.65
Case 2	13.3	12.6

Table A.2.: Expected and simulated Reynolds numbers for the two test cases.

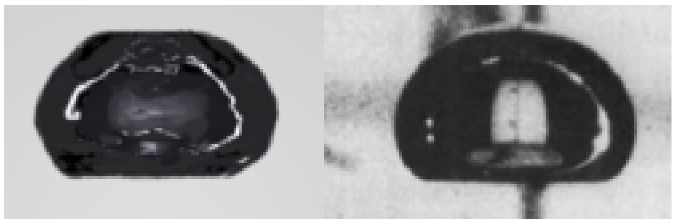

Figure A.2.: Simulation 1. Experimental results [1] (left - reproduced from Figure 3 (b), page 6) compared with simulated results (right).

Both simulations appear to match the experimental results fairly closely. The axis ratios and approximate shape of the bubbles are consistent with experiment, however the experimental results appear to have a much smoother outline than

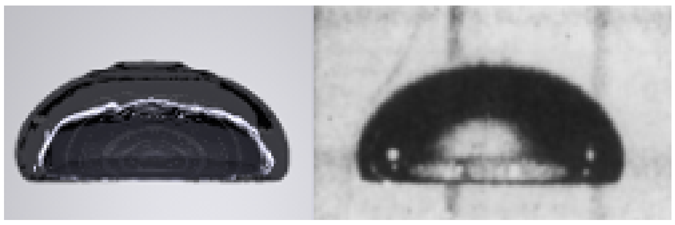

Figure A.3.: Simulation 2. Experimental results[1] (left - reproduced from Figure 3 (d), page 6) compared with simulated results (right).

the simulated results. In both cases it appears that the simulated results form more straight lines rather than smooth curves along the boundary. Of the two simulations, case one appears to suffer from this effect more. Potential causes of this error are discussed in Section 2.5.

B. Droplet Oscillation Frequencies

The power spectral densities for droplet oscillations, from which the frequencies presented in Section 4.3.3 are determined, are presented here. For each droplet three power spectra are documented. The first spectrum shows the frequencies of change in the ratio vertical axis length to the first horizontal axis length (X:Z ratio). The second shows the frequencies of change in the ratio vertical axis length to the second horizontal axis length (Y:Z ratio). The third shows the frequencies of change in the ratio of the two horizontal axes (X:Y ratio). The first two spectra are expected to show axisymmetric oscillations. The third spectrum is expected to show horizontal mode oscillations. All three spectra should show transverse oscillations.

While the values of the y-axes on the plots are not relevant, the relative values are, and are the same across all three plots. As there is interest over a wide range of frequencies the x-axis range has been set on a case-by-case basis.

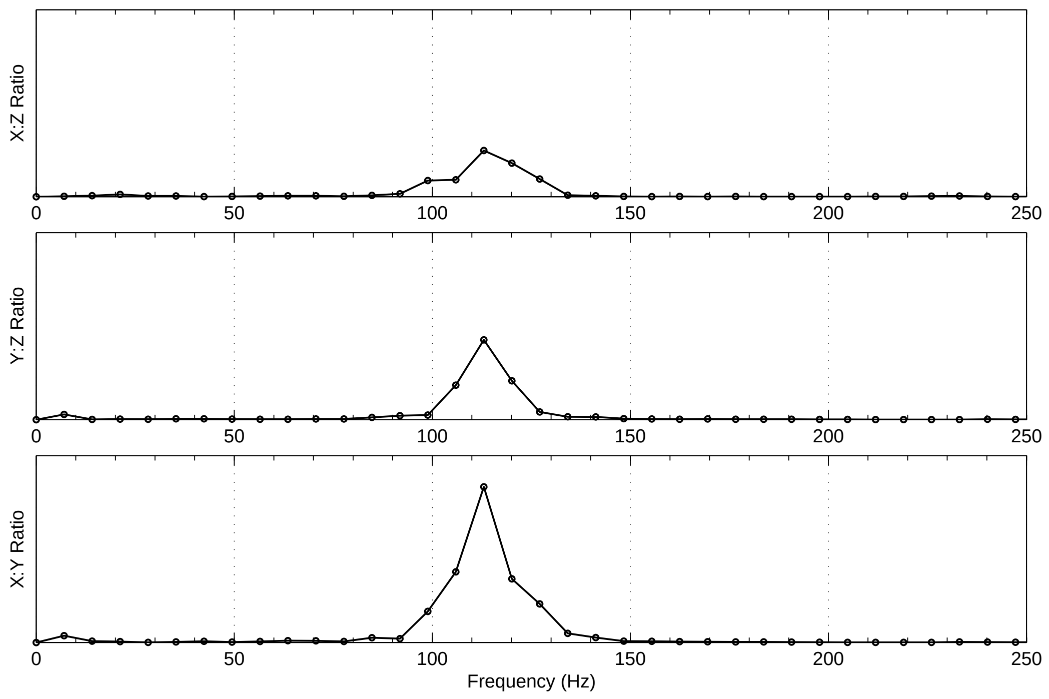

Figure B.1.: Axis ratio oscillation frequencies of the 1mm radius droplet.

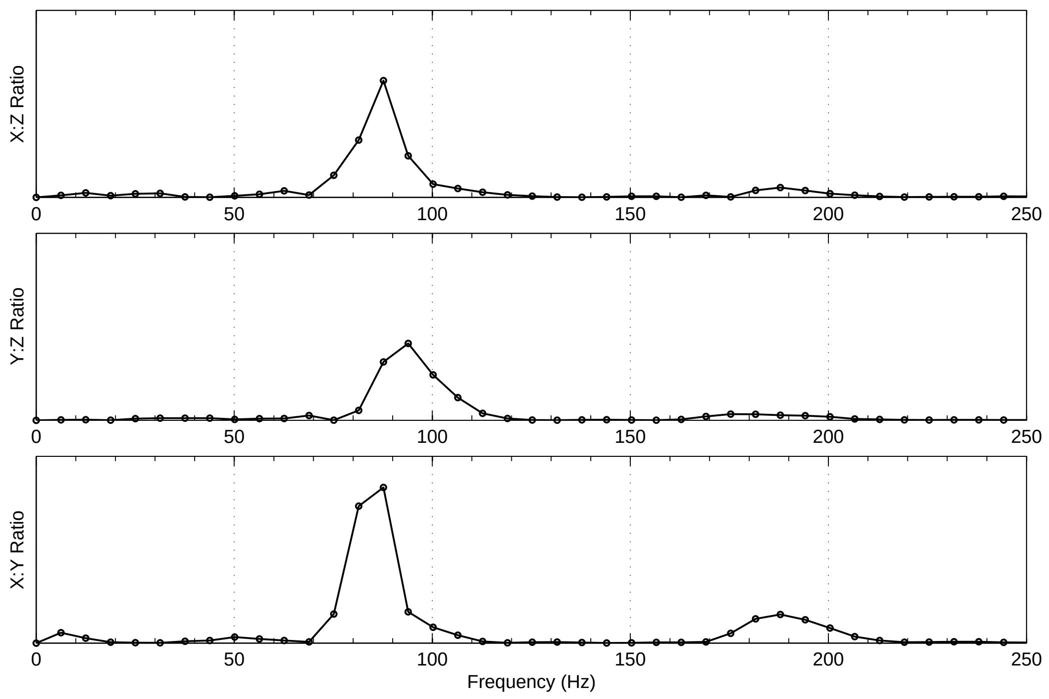

Figure B.2.: Axis ratio oscillation frequencies of the 1.25mm radius droplet.

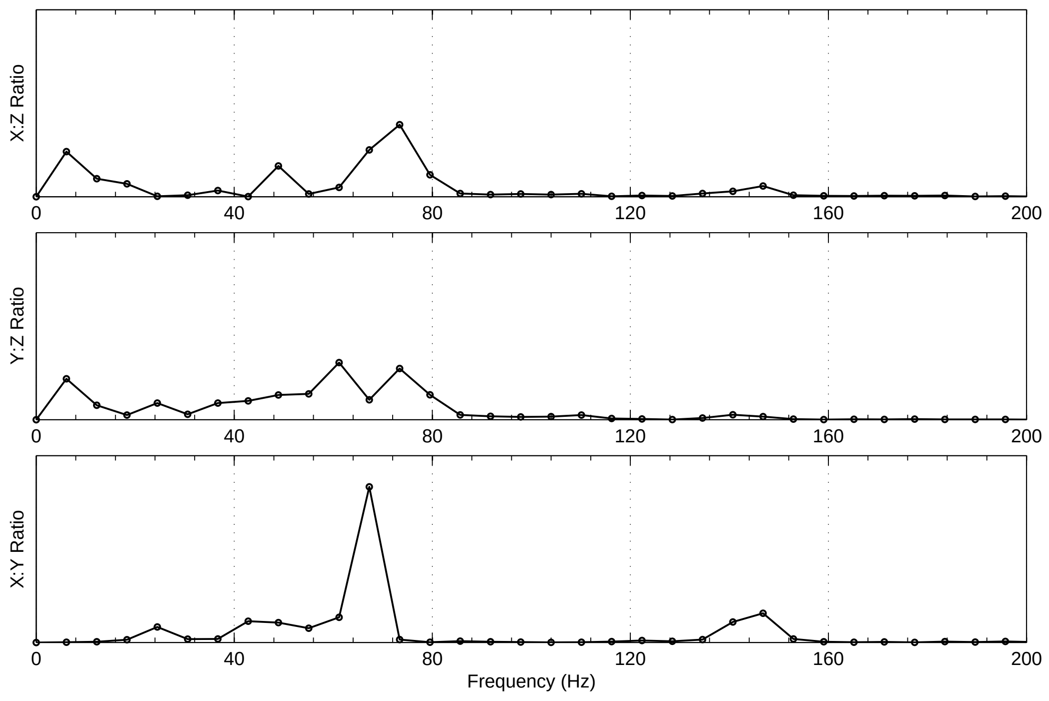

Figure B.3.: Axis ratio oscillation frequencies of the 1.5mm radius droplet.

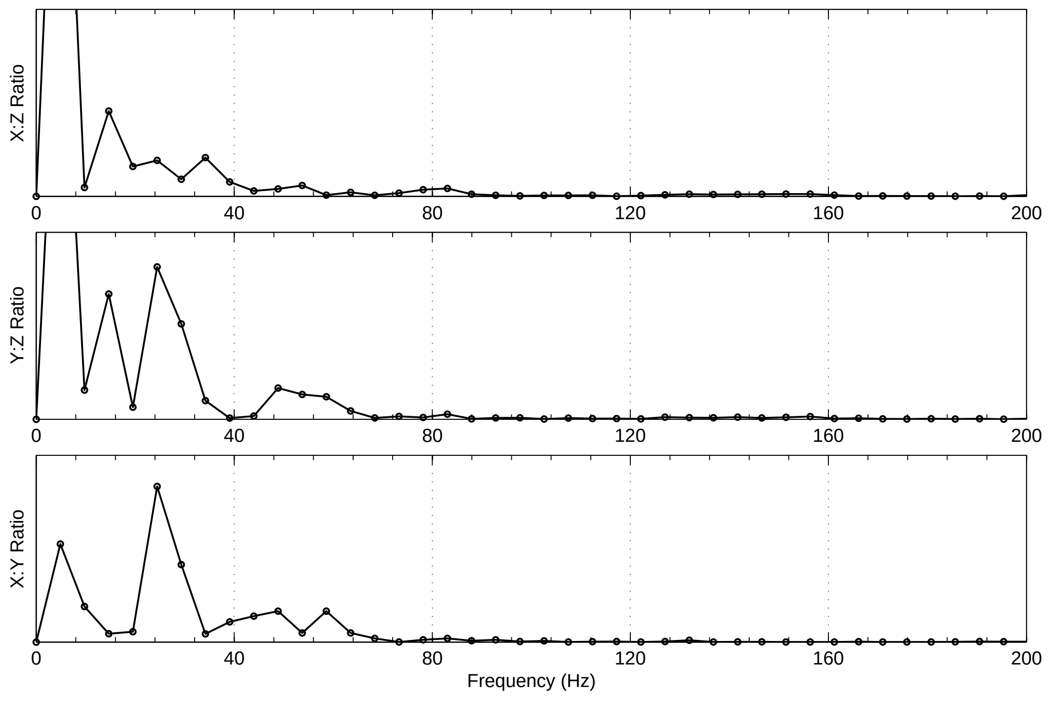

Figure B.4.: Axis ratio oscillation frequencies of the 1.75mm radius droplet.

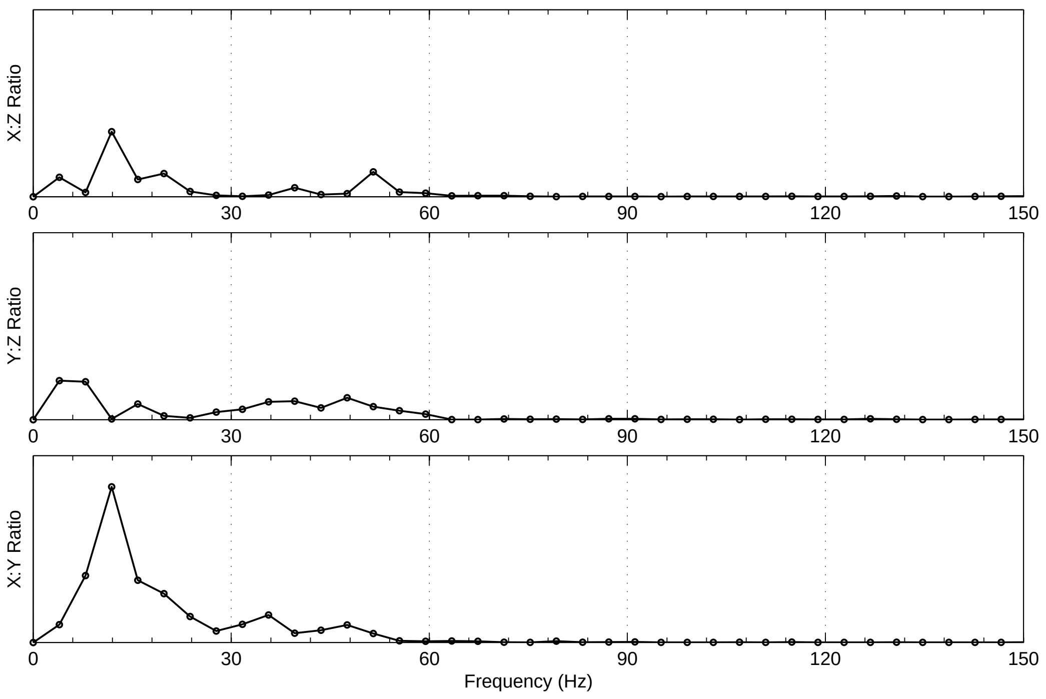

Figure B.5.: Axis ratio oscillation frequencies of the 2mm radius droplet.

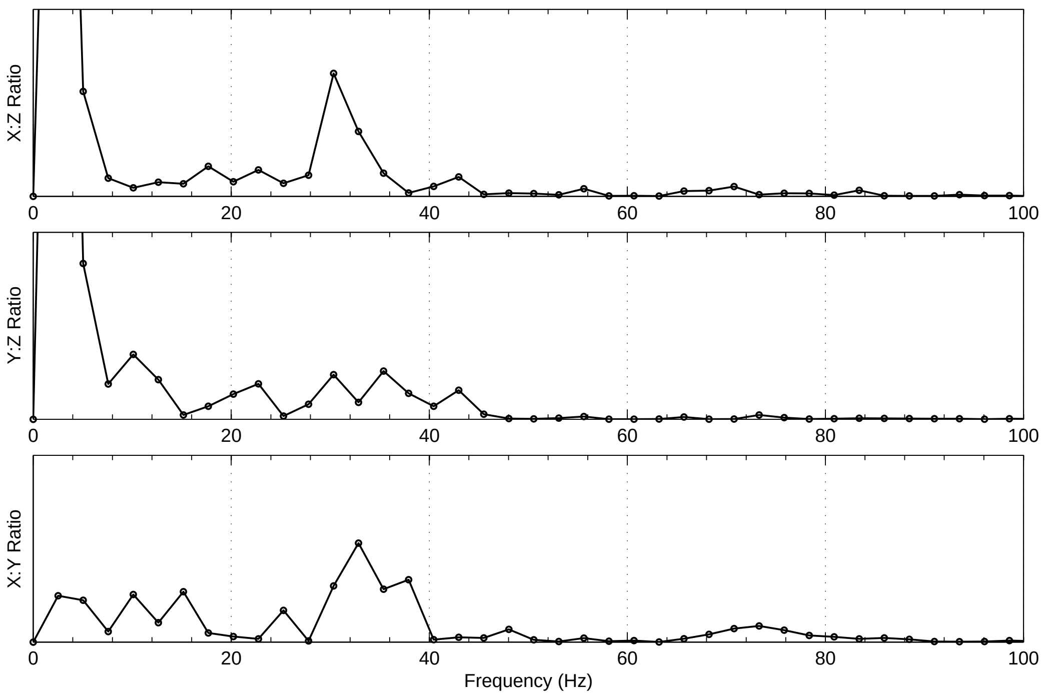

Figure B.6.: Axis ratio oscillation frequencies of the 2.25mm radius droplet.

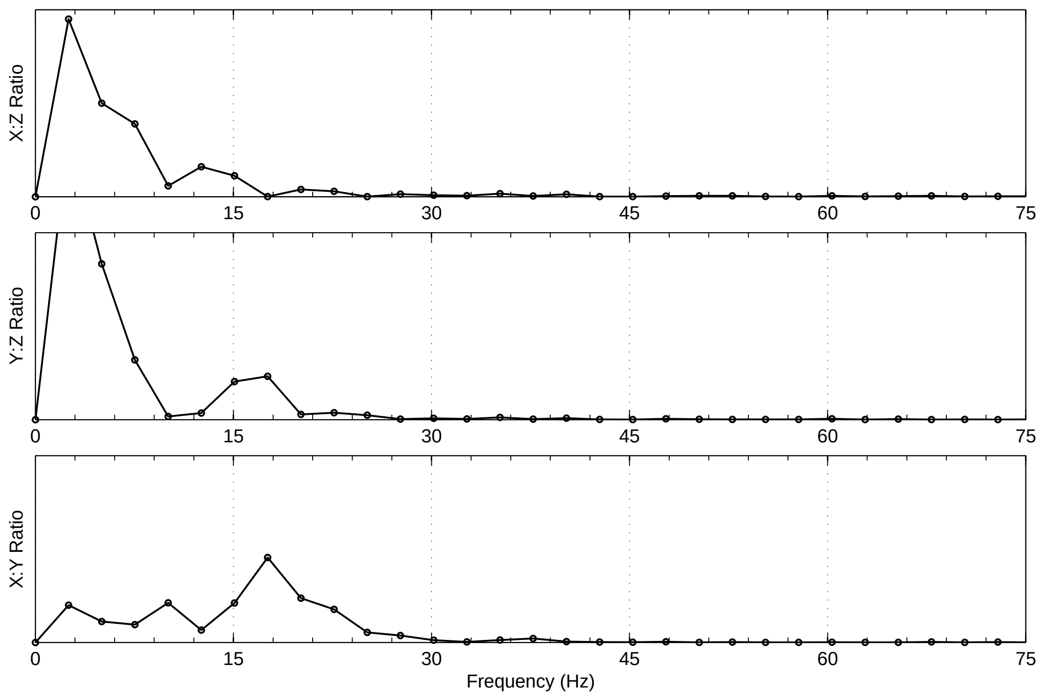

Figure B.7.: Axis ratio oscillation frequencies of the 2.5mm radius droplet.

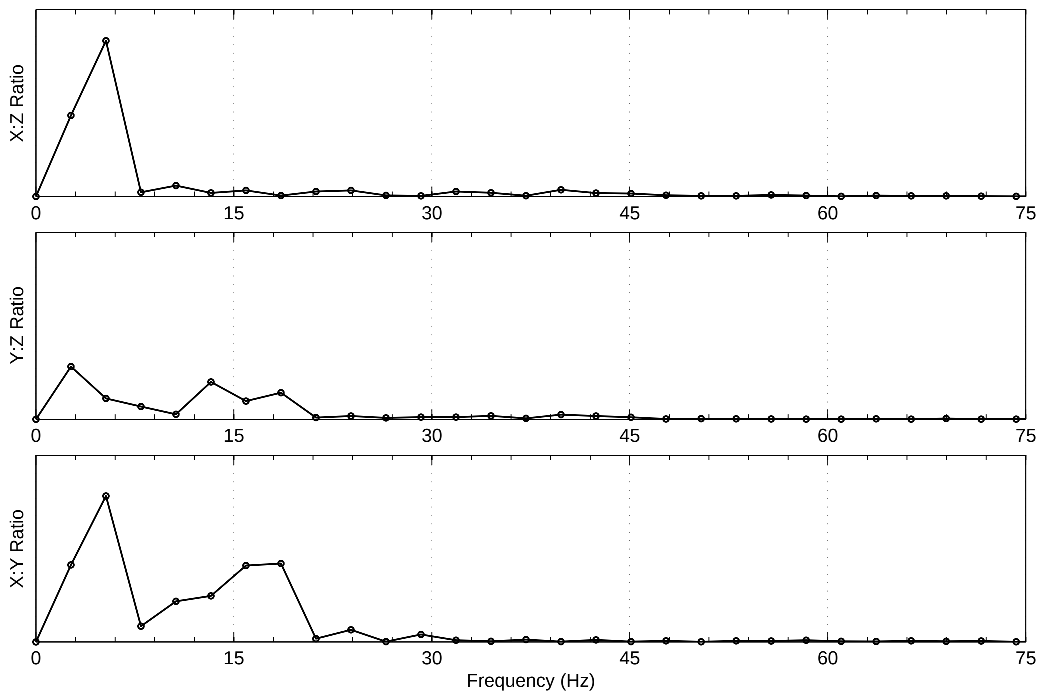

Figure B.8.: Axis ratio oscillation frequencies of the 2.75mm radius droplet.

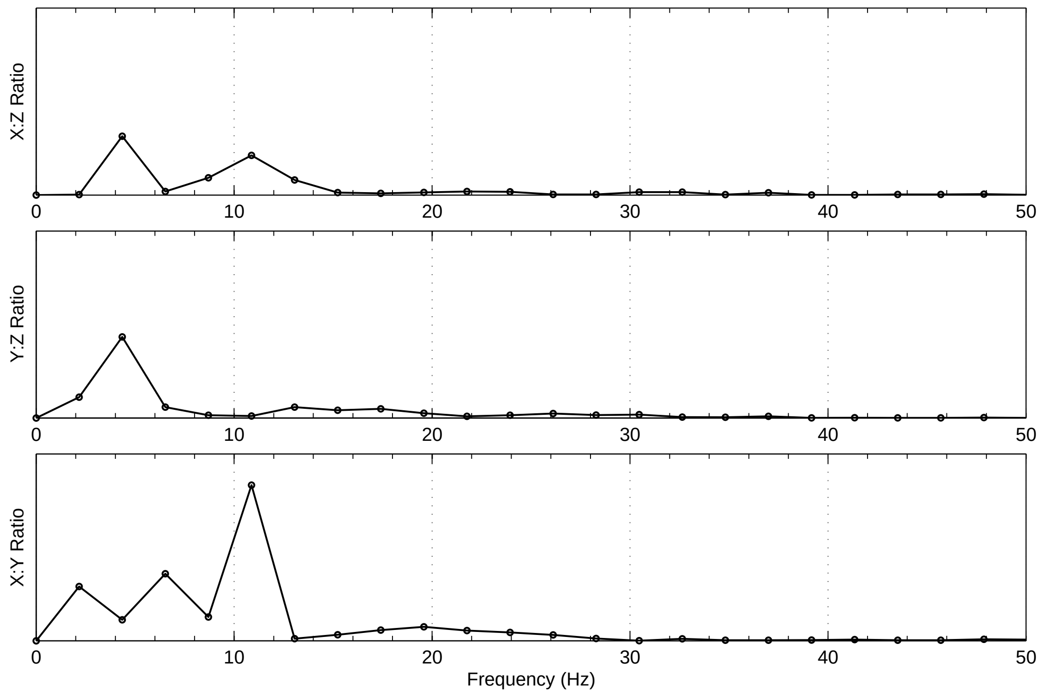

Figure B.9.: Axis ratio oscillation frequencies of the 3mm radius droplet.

C. Eddy Shedding Frequencies

The power spectral densities for eddy shedding in the droplets' wakes, from which the frequencies presented in Section 4.3.4 are determined, are presented here. For each droplet six power spectrums are documented, three for the near plane and three for the far plane. The entire plane spectrum is calculated from vorticity summed across the plane. This can be used to calculate the frequencies at which eddies are shed. The x and y axis ratios are calculated from the ratio of vorticities on planes splitting the respective axis. These frequencies can be used to determine if there is any structure to the positions that the eddies are shed at, and give some insight into that structure, should it exist.

While the values of the y-axes on the plots are not relevant, the relative values are. The two full plane spectra for a given plot have the same scaling, and the four ratio spectra for a given plot have the same scaling.

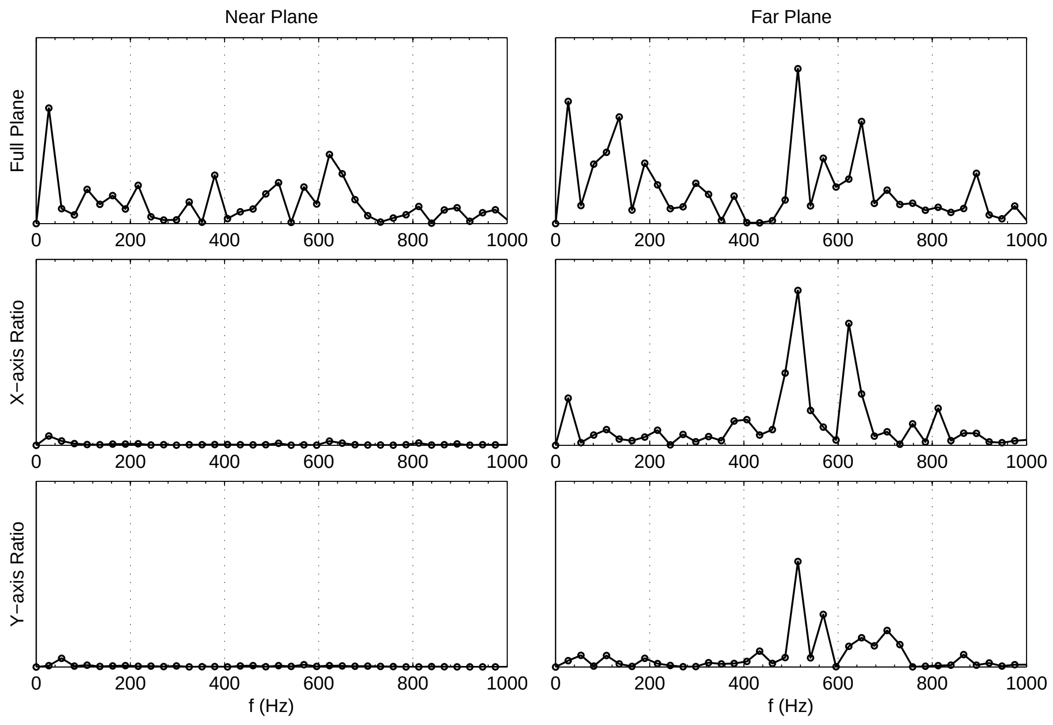

Figure C.1.: Eddy shedding frequencies of the 1mm radius droplet.

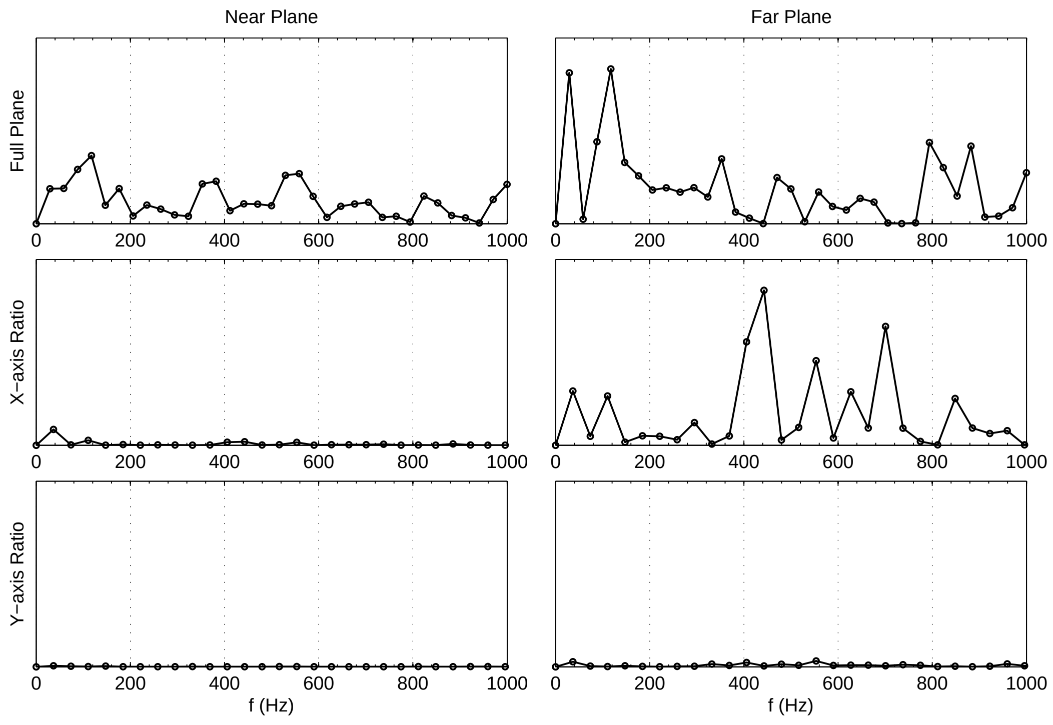

Figure C.2.: Eddy shedding frequencies of the 1.25mm radius droplet.

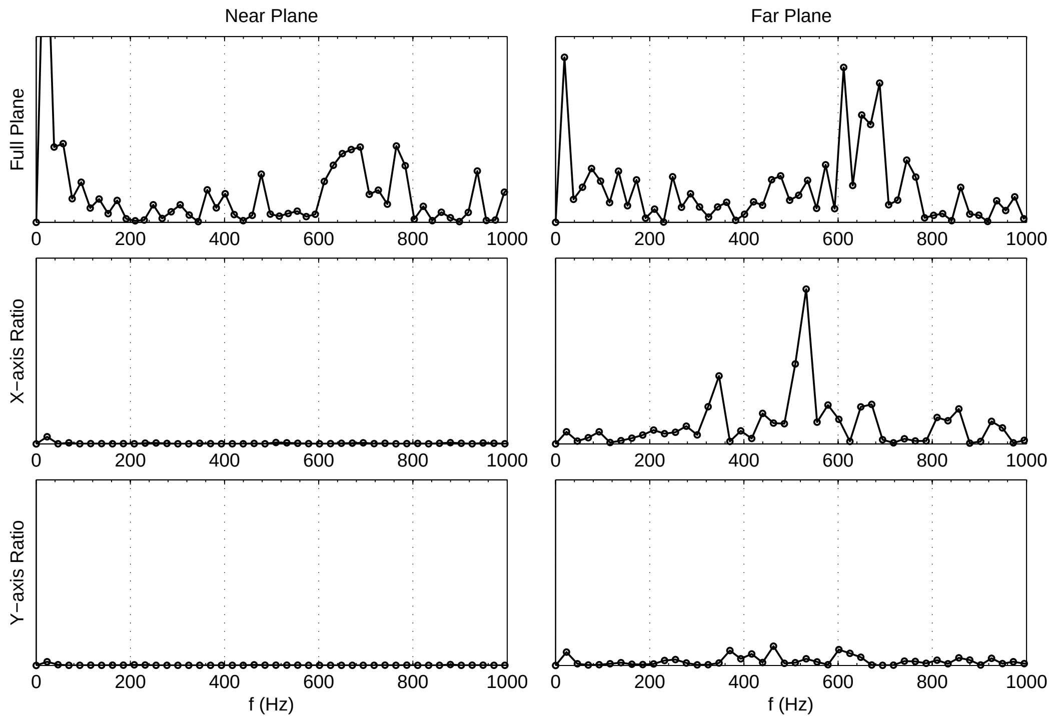

Figure C.3.: Eddy shedding frequencies of the 1.5mm radius droplet.

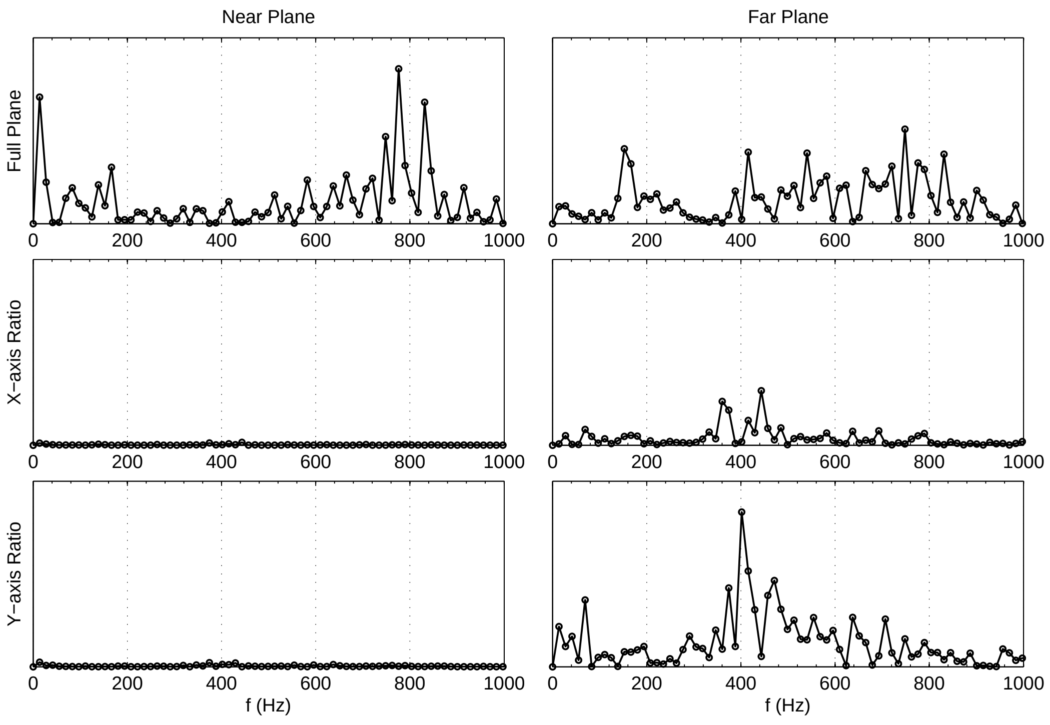

Figure C.4.: Eddy shedding frequencies of the 1.75mm radius droplet.

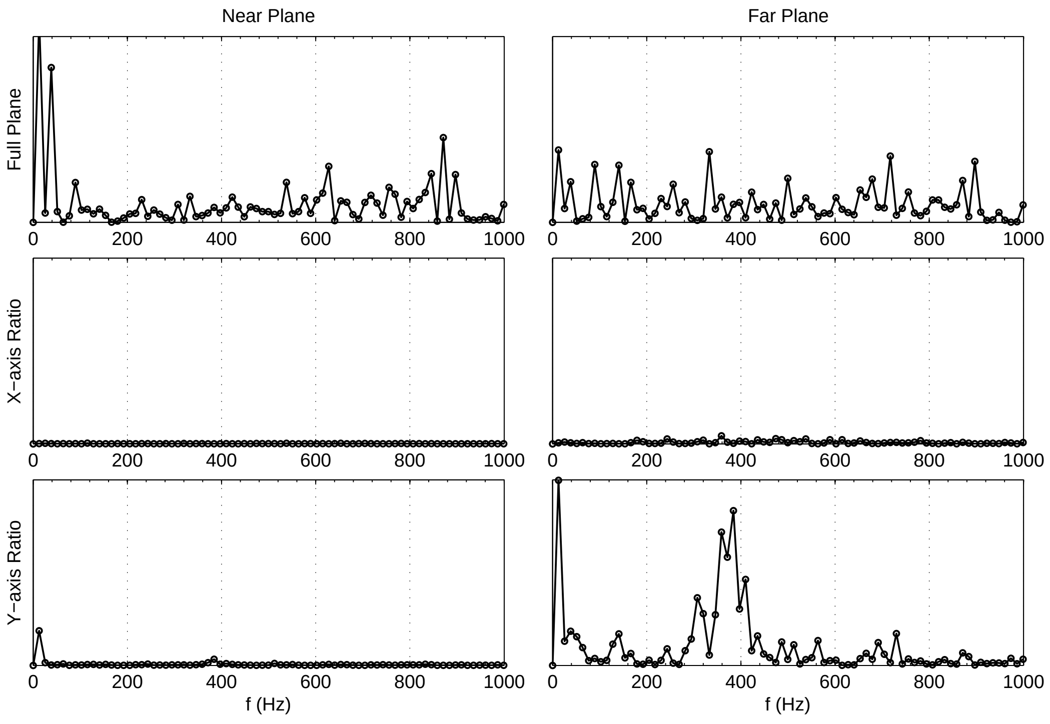

Figure C.5.: Eddy shedding frequencies of the 2mm radius droplet.

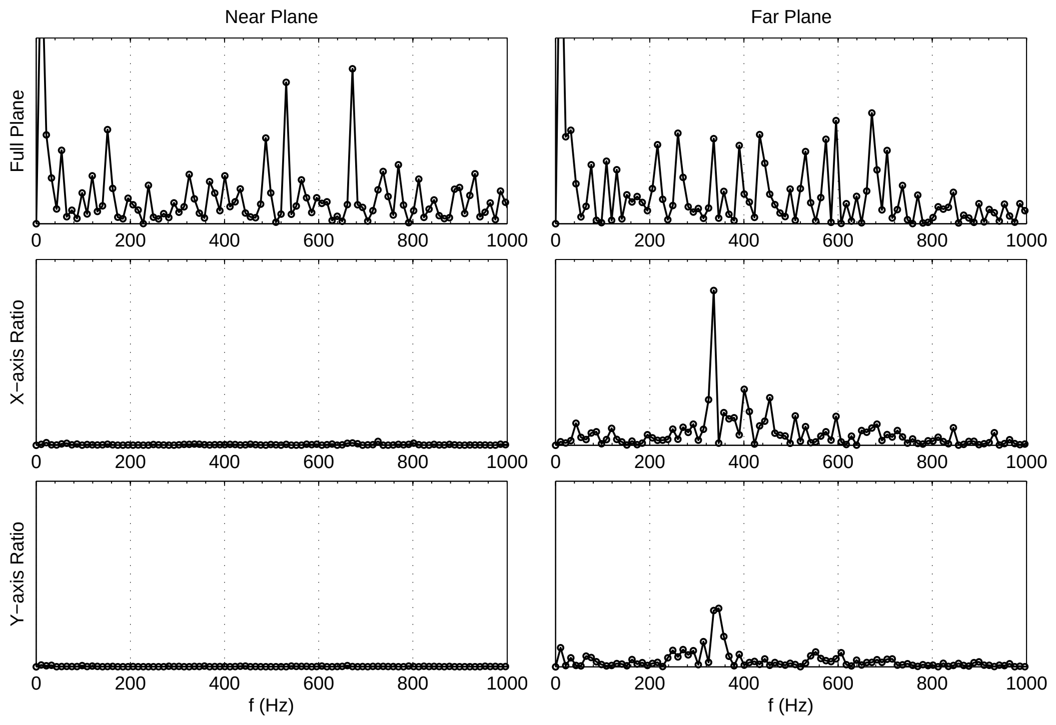

Figure C.6.: Eddy shedding frequencies of the 2.25mm radius droplet.

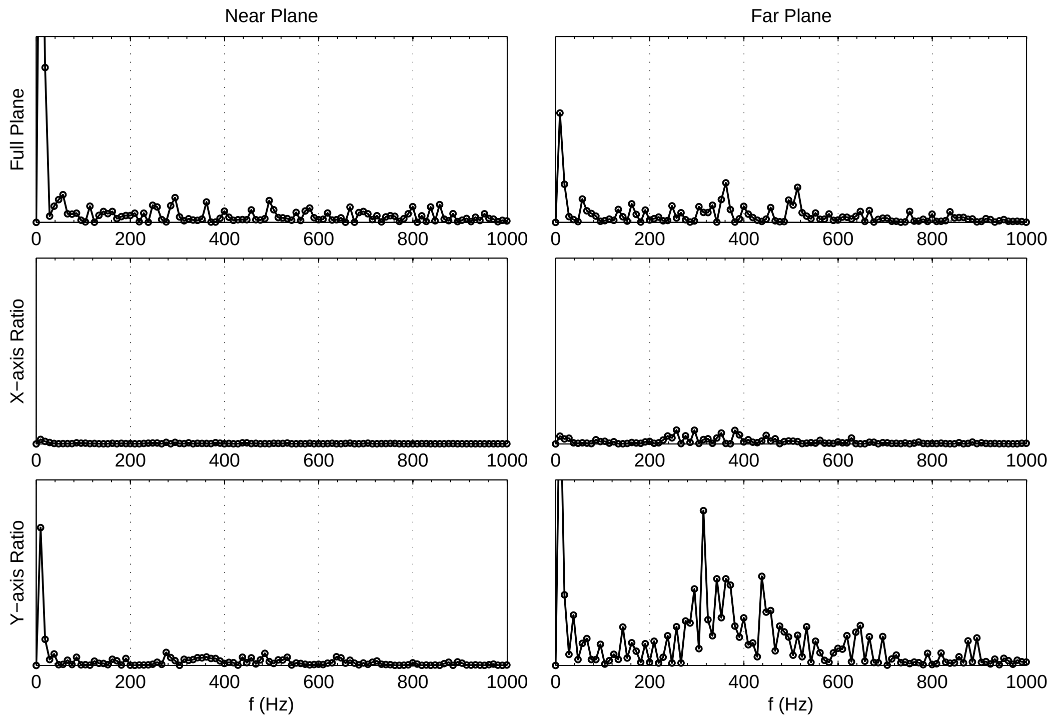

Figure C.7.: Eddy shedding frequencies of the 2.5mm radius droplet.

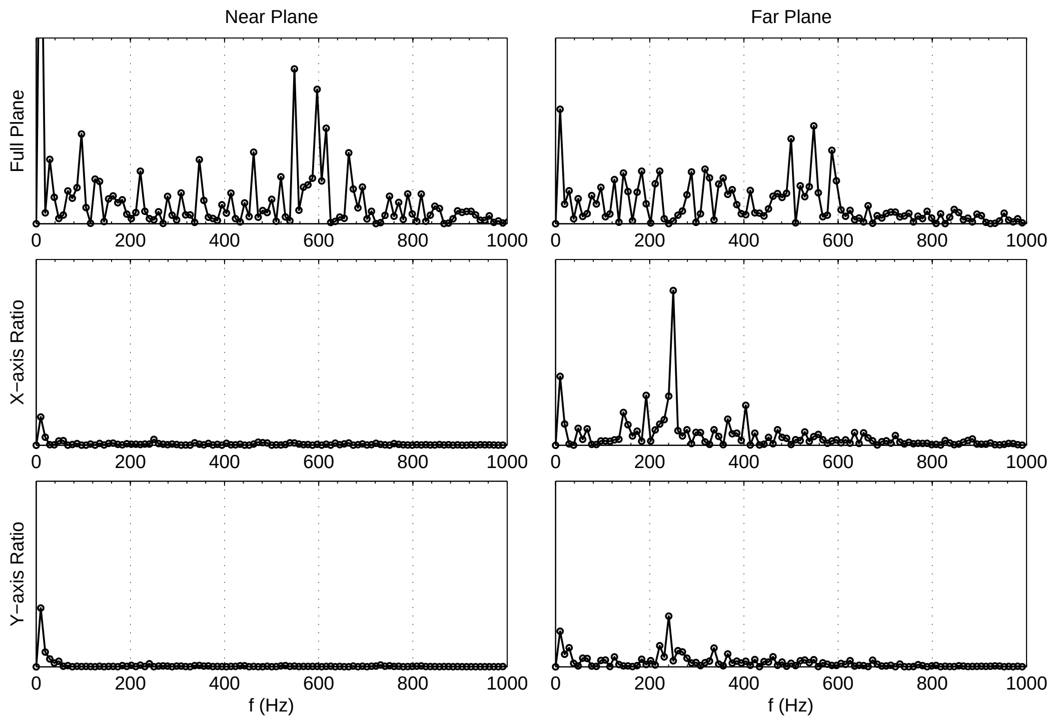

Figure C.8.: Eddy shedding frequencies of the 2.75mm radius droplet.

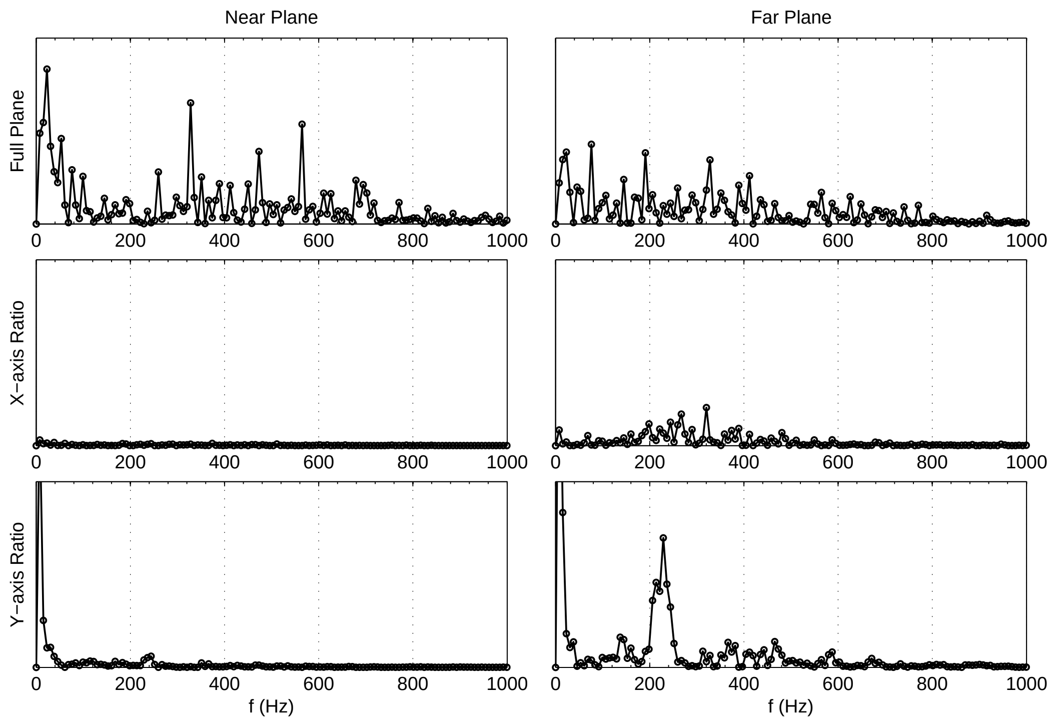

Figure C.9.: Eddy shedding frequencies of the 3mm radius droplet.