Simulation of the Ferrofluid Interface

by

Michael Honke

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

© Michael Honke 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Ferrofluids were initially invented as an additive for rocket fuels. The commercial applications of ferrofluids have since expanded, and have become popularized as desktop toys as well as an art form. Since 1987, ferrofluid simulations have been developed for engineering and physics applications. The Rosensweig instability, a visually appealing behaviour of ferrofluids, has been one of the focuses for ferrofluid simulation. While some simulations were successful, they have either placed restrictive assumptions on the problem, used non-physical models, or failed to reproduce this phenomenon due to high computational expense. One recent exception was a concurrent work from 2019 by Huang et al. that used a particle based fluid simulation method [1]. They successfully reproduced this phenomenon without these issues, but adopted a different computational approach.

We present a methodology for simulating ferrofluid with its accompanying Rosensweig instability using finite difference schemes within a grid based simulation. This is the first simulator to use a grid based methodology to approximately reproduce the Rosensweig instability. After Huang et al., our simulator is the second to approximately reproduce the Rosensweig instability with a nonrestrictive physically faithful model. The simulator accommodates any magnetic field and initial configuration of the fluid. Due to the high level of interface detail required by the Rosensweig instability we developed improved curvature estimation and surface tracking methods. We use a normal-aligned height function curvature stencil paired with a modified version of the particle level set. Instead of using particles for error detection, they are directly seeded on the interface to track it. These particles can then be used directly to determine the interface location for curvature estimation. The new particle level set is also able run on a GPU for a twenty to thirty times performance improvement compared to its CPU counterpart.

This coupling of methods produces curvature estimates that are two to five times more accurate than when operating independently of each other. After verifying the curvature and surface tracking methods, the ferrofluid simulator is demonstrated by inducing motion into a ferrofluid droplet using an applied magnetic field. Lastly, the Rosensweig instability is produced for a pool of ferrofluid sitting in a dish above a dipole magnet.

Acknowledgements

Regarding the choice to attend grad school, a mentor once told me: finding a good supervisor is critical, everything else is secondary. I want to thank Christopher Batty for being that supervisor, who fostered my scientific curiosity and guided me throughout this research project. His insightful perspectives on work-life balance, and research in academia and industry are invaluable lessons.

My committee members, Stephen Mann and Justin Wan, deserve recognition for their careful review of my thesis and subsequent helpful feedback. Outside of the committee duties, I want to thank Stephen for his excellent instructorship and both Justin and Stephen for the numerous engaging discussions.

I am grateful for the support and friendship of my fellow lab members and colleagues: Chufeng, Eddie, Henry, Jade, JC, Jonathan, Nathan, Ryan, and Yu. There was rarely ever a lonely moment. Our long discussions of research, work, and various other interests will be dearly missed. Outside the lab, I want to thank BH, David and Sashika for their continued friendship, interest in my research and long distance support.

The support, guidance, and time invested by my parents, Les and Lori, made embarking on this and past endeavours possible. I cannot thank you enough.

Dedication

To my loving parents,

Les and Lori Honke,

for your support and guidance.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Typical fluids are characterized by physical properties including but not limited to viscosity, surface tension, and density. Another physical property, called magnetic susceptibility, becomes relevant when studying fluids that contain dissolved ferromagnetic particles. Such fluids are called ferrofluids. In the absence of any magnetic fields ferrofluids behave as a conventional liquid. When a magnetic field is applied to a ferrofluid its high degree of magnetic susceptibility means it experiences a magnetic force [2]. This force can then be used to influence the movement of the fluid.

The ability to control a ferrofluid using magnetic fields meant they were once considered as a possible rocket fuel. In a zero gravity environment ferrofluids can be magnetically moved to prime a rocket motor [2]. Ferrofluids are commonly used in commercial applications including rotary shaft seals, such as those for computer disk drives [3], as well as a damping fluid for motors [4]. Loud speaker coils have used ferrofluids to manage heat, and provide damping while using magnetization to position the fluid [5].

Beyond engineering applications the way ferrofluids respond to an applied magnetic field creates several interesting visual phenomena. Complex labyrinthine patterns form when an immiscible mix of ferrofluid and non-magnetic fluid is trapped as a thin layer in a glass cell [3]. When a container of ferrofluid is exposed to a sufficiently strong magnetic field a pattern of peaks occurs on its surface, such as in Figure 1.1. This is called the normal field instability, also known as the Rosensweig instability, named after one of the original inventors of ferrofluids. This particular phenomena has captured the attention of artists. Mesplé, an artist who combines science and fine art, has used ferrofluids along with magnetic fields to create sculptures[1]. The Rosensweig instability has also spawned an

[1]http://www.mesple.com/mesple-technology-art

industry of desktop toys based on the interaction of contained ferrofluids with magnets[2].

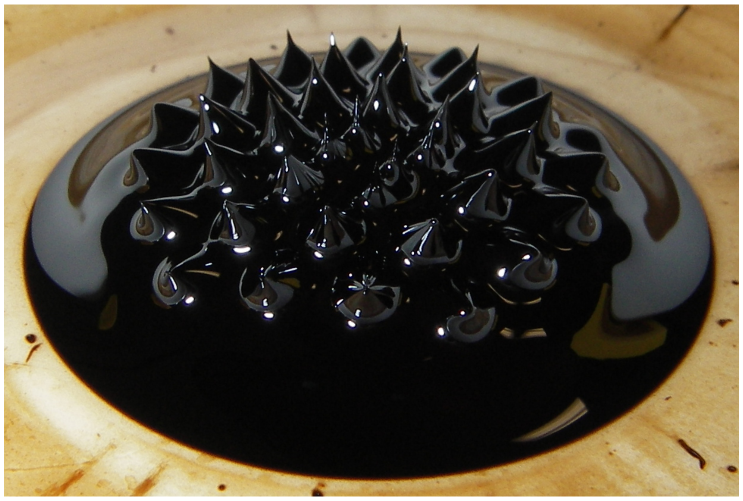


Figure 1.1: Example of a ferrofluid sitting in a dish with a magnet below exhibiting the Rosensweig instability.[3]

The interest in ferrofluids, both visually and for engineering applications, has motivated the development of various ferrofluid simulation techniques, which is the subject of this thesis. Simulators have been developed both within the fields of computational physics and more recently in computer graphics. Simulators such as these can be used by engineers to simulate how a ferrofluid might act in a particular application. Also artists can use simulations to plan and develop their next display. To accurately and easily represent a ferrofluid in computer graphics applications, such as in games or films, a physically based simulation is preferred to manual animation. However, despite the potential usefulness of ferrofluid simulations of the Rosensweig instability, there has been only limited success.

A pair of 2D and 3D ferrofluid simulators have been developed as part of this current work focused on recreating the Rosensweig instability. As far as this author is aware, the 3D simulator presented here is the first to use a grid-based fluid simulation to approximately simulate this instability, which includes characteristics such as peak growth and critical onset of peaking. Our simulator is the second such attempt, regardless of methodology, to use general conditions when simulating the instability. The first simulation to do so was recently published in July 2019, while our simulator was in development. However, this other simulator uses a particle based fluid simulation [1].

[2]https://www.czferro.com/

[3]Whats That? (64) CC BY 2.0 by jurvetson. Image was cropped.

Most fluid simulation techniques can be classified as either particle (Lagrangian) based methods, such as smoothed particle hydrodynamics (SPH), or grid based techniques, sometimes called Eulerian methods. There are also some hybrid methods such as particle in cell (PIC) and fluid implicit particle (FLIP) that use both particles and grids. There are a few advantages to using Eulerian methods. They have been shown to better resolve interface instabilities between two fluids [6]. Also, the underlying grid makes it easier to derive accurate finite differences [7], which are required for almost every method used in the simulation. Using finite differences then allows the use of popular linear algebra packages to calculate solutions for quantities such as pressure, viscosity and now also the magnetic field.

In the process of developing the present simulator we encountered multiple challenges related to representing the fluid's interface. Our response was to further improve several components of the basic grid-based fluid simulation framework. A massively parallelized modified particle level set method, where particles are used to track the surface directly, was developed. Additionally, we increased the accuracy of methods for measuring surface curvature which are then used to simulate surface tension. Lastly, we developed a specialized version of this surface curvature function that specifically integrates with our particle level set for further improved accuracy.

The contributions of our work is therefore two-fold. We demonstrate how grid-based simulation methods can be used to produce crucial features of the Rosensweig instability. However, our work extends beyond ferrofluid simulation. The particle level set method that we developed can be used for other simulations. In addition, we modified a curvature measurement method specifically for the aforementioned particle level set, producing a combination that improves accuracy.

Chapter 2

Previous Work

There has been limited success simulating ferrofluids for computer graphics applications until recently. However, in the field of computational physics the study of ferrofluids and their simulation has been more successful. However, as will be shown in the selection of previous works in these two fields, computational physics simulations tend to focus on simulating ferrofluids under specific conditions and assumptions, whereas graphics simulations aspire to simulate ferrofluids in a more general environment.

The first research done in ferrofluid simulation takes place in the field of computational physics. The earliest includes Rosensweig as an author in the work by Boudouvis et al. in 1987 [8]. They simulated the formation of peaks under static conditions using the Young-Laplace equations. They were able to replicate interface deflection trends with their simulation. However, their simulator did not simulate a fully 3D surface, but rather a heightfield. More recent examples include one from 2006 where Lavrova et al. simulated the Rosensweig instability under static conditions, along with more microscopic ferrofluid phenomena such as individual droplets [9] [10]. Like Boudouvis et al. their simulator also uses a heightfield, which limits the applications of the simulation. Also, they used the finite-element method (FEM), which requires generating a mesh, an expensive operation that necessitates special handling of topology changes. Gollwitzer et al. in 2007 detail their meshing process [11]. They strategically select their domain size to contain exactly one peak. This requires a priori knowledge of the expected peak geometry. As a result, the fluid configuration and applied magnetic field must be known before the simulation, thereby limiting user interaction with the ferrofluid. Using a similar methodology, FEM with height fields, Cao et al. in 2014 showed a simulation displaying a hexagonal pattern of peaks, characteristic of ferrofluids [12]. The main drawback of these methods is that they all make the assumption that fluid velocity is zero and has an interface representable by a heightfield, which while simplifying the simulation, limits them to static fluids without complex geometry.

In 2011, Yoshikawa et al. made the first attempt to simulate the Rosensweig instability under non-static conditions [13]. They used the FEM coupled with the moving particle semi-implicit (MPS) method, solving for the magnetic field and fluid respectively. The MPS particles in the fluid were used to create a mesh extending out into space to use the FEM. The magnetic forces were then calculated and applied to the fluid particles. This process repeats on each time step requiring the generation of many meshes over the course of a simulation. This constant conversion of data between MPS and FEM results in lengthy runtimes. Due to this high computational cost and some issues with volume conservation only one peak was generated over 24 hours of computation. No results with multiple peaks were presented.

The work in computational physics either does not focus on producing the Rosensweig instability under general conditions, or in reasonable runtimes for computer graphics applications. Ideally, such a simulator should simulate a dynamic ferrofluid, in 3D, with reasonable runtimes such that multiple peaks of the Rosensweig instability are produced. This type of simulation may also have applications in physics and engineering if it has sufficient accuracy.

The first such simulation seeking to address these requirements is from Ishikawa et al. in 2012 and 2013 [14] [15]. They used SPH fluid simulation, commonplace in graphics [16], with magnetic forces applied to the individual particles to simulate the main fluid body. This is unlike the previous works where the magnetic field is solved separately with respective forces being applied to the fluid in an iterative fashion. However this method did not produce peaks on its own. They proposed a couple of procedural models to add peaks to the fluid surface being simulated by SPH. While the models are based on equations from ferrohydrodynamic theory describing peak pattern and height for specific conditions, it is unable to handle peak formation for general scenarios.

In 2019, Huang et al. developed a particle based simulation [1] capable of generating peaks using a fully physical model of magnetic and fluid forces. Like Ishikawa et al. they use a SPH solver for their fluid. However, they employed a more sophisticated magnetism model, solving for the magnetic field separate from the fluid. One magnetic field calculation, using the fast multipole method, is done for every 10 time steps of SPH fluid simulation. Their simulator generates peaks without any post time step modifications, and follows physical trends related to parameters such as surface tension and the applied magnetic field. So far this is the most complete particle based ferrofluid simulation for general conditions.

As shown, particle based methods have received the only attention for ferrofluid simulation in computer graphics. However, there are other models that have been proposed that are capable of simulating a dynamic ferrofluid. In 2008 Afkhami et al. published such a model applying it to the study of ferrofluid droplets [17]. Their model has been used extensively since then for a number of other studies in computational physics [18] [19] [20], but not for simulating the Rosensweig instability. Critically, it does not assume that the fluid velocity is zero. It uses finite difference methods (FDM) on grids to solve for the magnetic field and the fluid. While the solves are done separately the shared data structure means that transferring magnetic forces back to the grid is not computationally expensive. The fluid is tracked using the volume of fluid method (VOF) avoiding the runtime cost associated with meshing.

The numeric ferrohydrodynamic model of Afkhami et al. serves as a suitable base for a general dynamic grid-based ferrofluid simulation. Therefore this thesis uses their method, albeit with some modifications. Curvature measurement is a concern due to its role in determining surface tension. This is an important component when attempting to produce the Rosensweig instability. Therefore, instead of using VOF, we reimplement their method using level set based surface tracking, which offers convenient calculation of smooth properties such as curvature [21]. We require a general 3D simulation, which has not been needed by Afkhami et al. as their simulations can use axisymmetric [17] [18] [19] or 2D domains [20]. A new discretization for a 3D Cartesian coordinate system is derived starting from their mathematical model.

With these modifications we present a simulator free of restrictive assumptions regarding fluid and magnet configuration, 3D and able to produce multiple peaks in a reasonable time. Multiple applications are possible with such a simulation. First for computer graphics, due to its comparatively computationally efficient methods, and secondly for engineering and physics due its physical basis.

Chapter 3

Physical Theory

The study of ferrohydrodynamics combines both fields of fluid dynamics and electromagnetic theory. The essentials of both subjects will be presented here. The actual discretizations and methods of solving for the fluid will be presented in the subsequent methods section (Chapter 4). Without any applied magnetic field ferrofluids resemble a typical fluid and are subject to the same physics.

3.1 Fluid Dynamics

The partial differential equation that describe the fluid flows of interest is called the incompressible Navier-Stokes equation

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla P = \nu \nabla \cdot \nabla \vec{u} + \frac{1}{\rho} \left(\sigma \kappa \hat{n} \delta_s + \vec{F_e} \right) \tag{3.1}$$

where $\vec{u}$ is the velocity, ρ is the density, P is pressure, $\vec{F_e}$ are external forces due to gravity and/or magnetism and ν is the kinematic viscosity coefficient. The $\sigma \kappa \hat{n} \delta_s$ term is responsible for the surface tension force with σ being the surface tension coefficient and κ the mean curvature of the fluid interface. The force is along the surface normal $\hat{n}$ and only applied on the interface as indicated by Dirac delta function δ_s. The velocity is subject to a divergence-free condition

$$\nabla \cdot \vec{u} = 0 \tag{3.2}$$

which enforces incompressibility.

To optimize the solving of Equation 3.1 the common technique of splitting is used instead of attempting to solve the entire equation at once [7]. This breaks the equation first into an advection part

$$\frac{D\vec{u}}{Dt} = 0 \tag{3.3}$$

where $\vec{u}$ can be replaced by another quantity if one exists that requires advection such as colour or smoke density. The material derivative is defined as follows.

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla \tag{3.4}$$

Next, a body forces part

$$\frac{\partial \vec{u}}{\partial t} = \frac{\vec{F}_e}{\rho} \tag{3.5}$$

which is used to apply the forces due to gravity and magnetism as a velocity update. Then there is the viscosity part

$$\frac{\partial \vec{u}}{\partial t} = \nu \nabla \cdot \nabla \vec{u} \tag{3.6}$$

which is due to internal friction in the fluid. The accumulation of new velocities will result in a pressure change. The following equation

$$\frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho} \nabla P = 0 \tag{3.7}$$

calculates the associated pressure due to the updated $\vec{u}$ values. This equation is discretized and solved in conjunction with Equation 3.2. A final velocity update is then done. At this point in the process, the velocities are divergence-free. This pressure solve requires a careful treatment of interface conditions when near the surface due to surface tension which will be discussed in Section 4.5.

These steps are repeated in the order presented to simulate a fluid. Critically, the only time the velocities are divergence-free and hence valid is immediately after the pressure solve. The divergence-free condition is no longer true after advecting the velocities in their own field [7]. Advection through a diverging or converging vector field can cause material to be erroneously created or deleted. Therefore all other quantities are advected before the velocities are.

3.2 Ferrohydrodynamics

Having reviewed fluid properties, the additional magnetic properties that define a ferrofluid can now be presented. First the chemistry of a ferrofluid is reviewed to understand how it is magnetic. Then some general magnetic theory is presented and isolated to the cases applicable for ferrofluids giving the main equation that will be solved to find the magnetic potential. Lastly, some theory regarding the macroscopic behaviour of ferrofluids related to the Rosensweig instability is discussed. This can later be used to verify if a simulation is performing as expected.

3.2.1 Chemistry

The simplest type of ferrofluid would be a homogeneous one. This type of fluid has not been synthesized and remains only possible in theory [3]. In fact ferrofluids are not naturally occurring and must be carefully synthesized as heterogeneous fluids. They are typically colloidal suspensions of small (3-15nm) solid magnetic particles in water or oil. The individual particles are coated in a surfactant. It is a single layer coating of molecules that are attracted to the magnetic particle while being compatible with the solvent of the ferrofluid.

A usable ferrofluid is stable against a number of forces that seek to precipitate the magnetic particles out of it. They are summarized here but are presented in more detail by Rosensweig [3]. First, the force of gravity pulls the particles downwards. However, since the magnetic particles are small the thermal motion, specifically Brownian motion, is sufficient to keep them dispersed in the fluid. Since the particles are magnetic they can become stuck together if they become too close (due to electromagnetic force strength being inversely related to separation), which can randomly happen as they travel within the carrier fluid. If the particles are kept sufficiently small then thermal motion is energetic enough to separate them again. Similarly the particles can be attracted to each other through electrical forces called the van der Waals force. This is due to electron interactions between two magnetic particles. Steric hindrance, due to the surfactant coating, is necessary to keep these forces from growing too strong and overpowering Brownian motion. Lastly, and an important consideration for practical use, is what happens when a strong magnetic gradient is applied to the ferrofluid? The magnetic particles will clump together near the source of the gradient despite Brownian motion attempting to redistribute the particles. Figure 3.1 depicts this happening for a basic ferrofluid experiment setup. Steric hindrance keeps the particles sufficiently separated. When the magnetic field source is removed Brownian motion is able to redistribute the particles without them becoming permanently clumped together due to the previously mentioned interparticle forces.

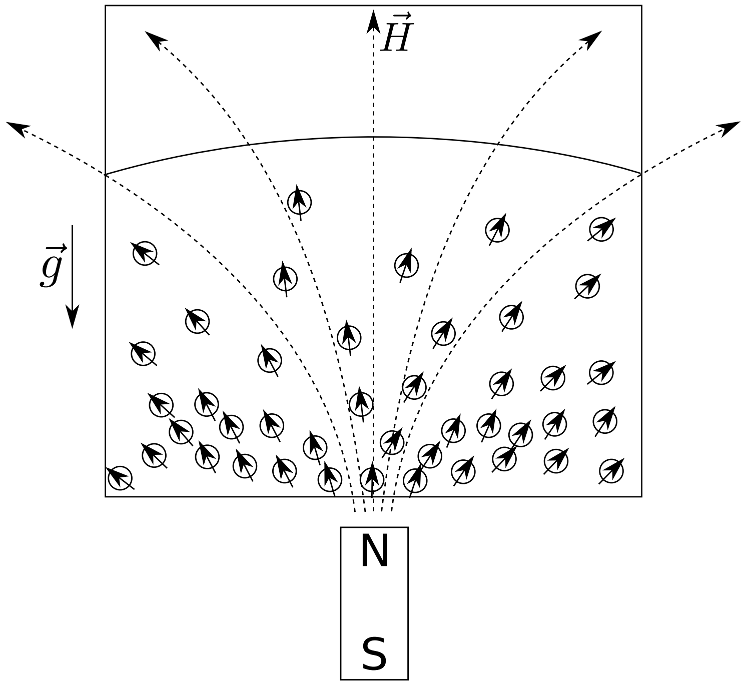


Figure 3.1: A basic ferrofluid experiment. The magnet is applied at the bottom of the ferrofluid container. Particles clump towards the magnetic source. Since they are ferromagnetic their average magnetic dipoles (represented by arrows) align almost perfectly with the applied field $\vec{H}$.

The compositional requirements of a ferrofluid leave it electrically insulating. It can be safely assumed that there are no electrical currents within the fluid. This conveniently simplifies the electromagnetic theory to just magnetic theory.

3.2.2 Magnetic Theory

The ferrofluid starts experiencing magnetic forces when an external magnetic field is applied. The individual particles become magnetized resulting in complex self-interacting forces. To attempt to simulate this process requires creating magnetic fields that obey the following physical laws. The first restriction is due to Gauss's law for magnetism

$$\nabla \cdot \vec{B} = 0 \tag{3.8}$$

where $\vec{B}$ is the magnetic induction, measured in Tesla (T). The magnetic induction is dependent on the properties of the surrounding material expressed by

$$\vec{B} = \mu \vec{H} \tag{3.9}$$

where μ is the permeability of the respective material and $\vec{H}$ is the magnetic field measured in amperes per meter ($\mathrm{A\,m^{-1}}$). The permeability μ is defined by

$$\mu = \mu_0(1 + \chi) \tag{3.10}$$

where χ is called the magnetic susceptibility. The magnetic susceptibility is zero in a vacuum, hence $\mu = \mu_0$ where μ_0 is the magnetic permeability of free space.

The magnetic field is restricted by Ampère's law

$$\nabla \times \vec{H} = J_f + \frac{\partial D}{\partial t} \tag{3.11}$$

where J_f is the free current and D is the electric displacement field [22]. However, in the absence of electric charge Equation 3.11 simplifies to

$$\nabla \times \vec{H} = 0 \tag{3.12}$$

which is a safe assumption due to the chemistry of ferrofluids [3].

Next, we introduce the concept of magnetic potential. Magnetic potential is defined as

$$\vec{H} = -\nabla \Psi \tag{3.13}$$

and is called the scalar magnetic potential, which is only applicable when $\vec{H}$ is a conservative (or irrotational) field as shown by Equation 3.12. Using this relationship and Eq. 3.8 gives

$$\nabla \cdot (\mu \nabla \Psi) = 0 \tag{3.14}$$

allowing for Ψ to be solved for some given boundary and interface conditions. If μ is assumed to be constant over all space then further simplifications can be made giving

$$\nabla^2 \Psi = 0 \tag{3.15}$$

which is Laplace's equation. The numerical discretization of this equation is straightforward and common. However Equation 3.15 is not valid for ferrofluid simulations because μ cannot be assumed constant and removed from the equation. Using the product rule Equation 3.14 becomes

$$\mu \nabla^2 \Psi + (\nabla \mu) \cdot \nabla \Psi = 0 \tag{3.16}$$

which more clearly presents the effect of a non-constant μ. The air around a ferrofluid is essentially non-magnetic $\mu_a = \mu_0$, which will be an assumption used throughout this work. However in a magnetically susceptible ferrofluid $\mu_f \neq \mu_0$. At the interface between the ferrofluid and air this non-constant μ results in $\nabla\mu \neq 0$. Furthermore, μ_f may not be constant even within the ferrofluid. Equation 3.14 cannot be further simplified and is called a varying coefficient Poisson equation. This particular case is non-trivial to solve as there are various methods of treating the discontinuity of μ at the air-fluid interface.

Within the fluid μ_f can vary. Some simulators make the assumption that μ_f is constant, but this assumption disregards the chemistry of the fluid. When a magnetic field is applied the distribution of the magnetic particles in the fluid changes as discussed in Section 3.2.1. The particles are responsible for the magnetic susceptibility of the fluid and as a result μ_f is a function of the magnetic field strength. The experimentally verified [17] and theoretically derived [3] relationship for the magnetization is

$$\vec{M}(\vec{H}) = M_s\mathbf{L}\left(\frac{\mu_0 m|\vec{H}|}{k_b T}\right)\frac{\vec{H}}{|\vec{H}|} \tag{3.17}$$

where $\mathbf{L}$ is the Langevin function $\mathbf{L}(\alpha) = coth(\alpha) - \alpha^{-1}$ and $\vec{M}$ is the magnetization of the fluid. Magnetization is the magnetic dipole moment per unit volume of the fluid [22]. The total magnetization is the macroscopic result of many small magnetic domains aligning or anti-aligning with each other. Magnetization also varies with the magnetic field according to Equation 3.18.

$$\vec{M} = \chi\vec{H} \tag{3.18}$$

Using Equations 3.10, 3.17 and 3.18 together give

$$\mu_f = \mu_0\left(1 + M_s\mathbf{L}\left(\frac{\mu_0 m|\vec{H}|}{k_b T}\right)\frac{1}{|\vec{H}|}\right) \tag{3.19}$$

which can be used to obtain μ_f for the fluid containing regions when solving Equation 3.14. Alongside μ_0 the other physical constant is Boltzmann's constant k_b. The remaining parameters are fluid-specific. M_s is the saturation magnetization, T is the temperature and m is the total magnetic moment of each particle.

3.2.3 Macroscopic Description of the Rosensweig Instability

The following is a conceptual explanation of why the Rosensweig instability occurs. Consider a flat ferrofluid surface with a sufficiently strong magnetic field applied perpendicularly to it. Thermal motion, environmental vibrations and other disturbances cause small random deformations to occur on the fluid's surface. Magnetic flux concentrates at the top of these deformations, and decreases towards the bottom of them [3]. The deformations then continue to grow creating a feedback loop as they alter the total magnetic field. Their growth is limited by surface tension and gravity reaching an equilibrium with the magnetic forces [13].

Rosensweig has previously derived, using ferrohydrodynamic theory, a set of equations that describe the empirical properties of the instability. To facilitate the derivation a number of assumptions are made. For example, the ferrofluid is assumed to be infinite, inviscid and in a vacuum [3]. However, the resulting equations are still useful for verification purposes since they show the trends that should be observed in a simulated ferrofluid. Trends such as these were used by Ishikawa et al. to procedurally generate peaks on their ferrofluid surface [14] [15]. As previously explained, using these trend equations restricts the output of the simulator to just the conditions those equations can describe. Besides using Equation 3.19 to avoid simulating a ferrofluid at a molecular level, our simulation uses only the Navier-Stokes equation (Equation 3.1) and the variable coefficient Poisson magnetic potential equation (Equation 3.14) to model the ferrofluid. The following equations are only used to check if our simulations have physically correct behaviour.

In addition to the previously mentioned assumptions, the trend equations [23] are derived assuming that the ferrofluid peaks can be represented as a height field

$$z_0 = \hat{z}_0 \mathrm{Re}\left(e^{i(\omega t - \vec{k}\cdot\vec{x})}\right) \tag{3.20}$$

where ω is the frequency and $\vec{k}$ is the wavevector. Also assumed is that the magnetic field and gravity are applied perpendicularly to the flat interface of the ferrofluid. Such a predefined setup limits the usefulness of the trend equations to quantitatively verify a general ferrofluid simulation. However, for simple simulations the conditions should be similar enough that qualitative trends should hold true.

The critical magnetization M_c specifies the minimum magnetization M that is required to induce the onset of the Rosensweig instability [23]:

$$M_c^2 = \frac{2}{\mu_0}\left(1 + \frac{\mu_0}{\mu}\right)\sqrt{\rho g \sigma} \tag{3.21}$$

indicating that as gravity, fluid density and surface tension are increased then intuitively the fluid must have its magnetization increased to generate the Rosensweig instability. The magnetization can be increased through using a stronger applied field or increasing the saturation magnetization. The M_c as defined is a lower limit to establish a stable pattern of peaks. M_c could be higher due to the effect of boundaries which are not considered here. Regardless, there exists a M_c such that an accurate simulation should not generate peaks with $M < M_c$, and can start generating peaks when $M \geq M_c$. The critical field is derived from the critical magnetization and is also a useful parameter [24].

$$H_c = \left(\frac{2}{\mu_0} \frac{\mu_0/\mu + 1}{(\mu_0/\mu - 1)^2} \right)^{1/2} (\rho g \sigma)^{1/4} \tag{3.22}$$

After peaks initially form they can further develop. First define ε as the bifurcation parameter

$$\varepsilon = \frac{H^2 - H_c^2}{H_c^2} \tag{3.23}$$

which is the relative strength of the externally applied magnetic field H. Here $\varepsilon = 0$ indicates the critical magnetic field for peak formation. Then the peak height h (for hexagonal patterns) is given by

$$h = A \frac{b(1+\varepsilon) + \sqrt{b^2(1+\varepsilon)^2 + 4a\varepsilon}}{2a} \tag{3.24}$$

where a, A and b are used to fit the experimental data [12]. Intuitively the peak height increases with additional magnetic field strength past the critical value required for peak formation ($\varepsilon > 0$). The equations presented here are later referred to in Chapter 6 when discussing our simulated version of the Rosensweig instability.

Chapter 4

Methods for Fluids

The methods used to implement the simulator based on the physics presented in Section 3 are detailed here. The details of the fluid solver are presented first since the magnetic forces are added in as an external force later. However, the fluid solver was not designed in isolation, but to specifically accommodate a ferrofluid's features, such as the Rosensweig instability. Therefore, such a solver would probably work well for other fluids that require the ability to produce detailed static structures on their interface. A brief overview of the simulation steps is provided in Figure 4.1.

4.1 General Simulation Data Structures

Our simulator is grid-based instead of being particle-based. While less common for the study of ferrofluids, grid-based methods are common in other applications and this simulator shares the same main data structures referenced by Bridson [7].

4.1.1 The MAC Grid

A staggered marker-and-cell (MAC) grid is used to store cell-centred and face-centred (or edge-centred in 2D) data on the grid. The distance between adjacent cell-centres is represented with Δx. For n_x grid cells, $n_x \Delta x$ gives the total size of the simulation domain along the respective dimension. Each node is assigned an index (i, j, k). The faces of the cells are referenced by using a half-index in the direction relative to their cell-centre. For

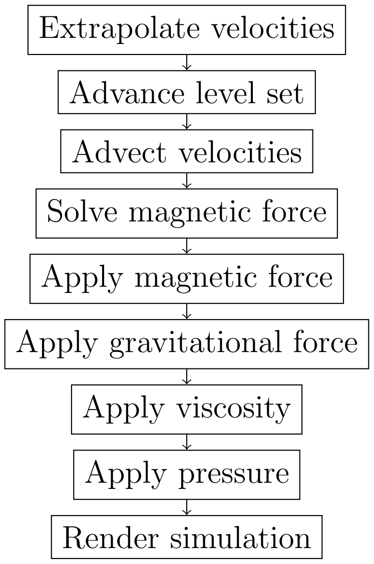


Figure 4.1: Overview of one simulation time step.

example, the edges of the cell (i, j) are then located at $(i+1/2, j)$, $(i-1/2, j)$, $(i, j+1/2)$ and $(i, j-1/2)$. Refer to Figure 4.2 for a 2D example.

Cell-centred data includes pressure and level set values (discussed in Section 4.3.1), whereas velocities are stored on the faces. It becomes apparent then that there is no clear velocity vector $\vec{v}$ since each component is stored separately. However, interpolation can be used to create a velocity vector anywhere on the grid. At the face-centred points the interpolation is simply the average of the four surrounding values of the other velocity component. Since for n cells there are $n+1$ faces along a particular dimension, each velocity component has its dimensions increased by one along the direction it is pointing relative to the dimensions of the cell values. Each cell also has an assigned material type label. For this simulation there are three labels, one each for air, fluid and solids.

While such a data structure seems unintuitive and perhaps error prone due to the half-indexing and interpolation required, it naturally facilitates centred difference approximations of PDEs which are used for numerically solving the pressure and magnetic field equations.

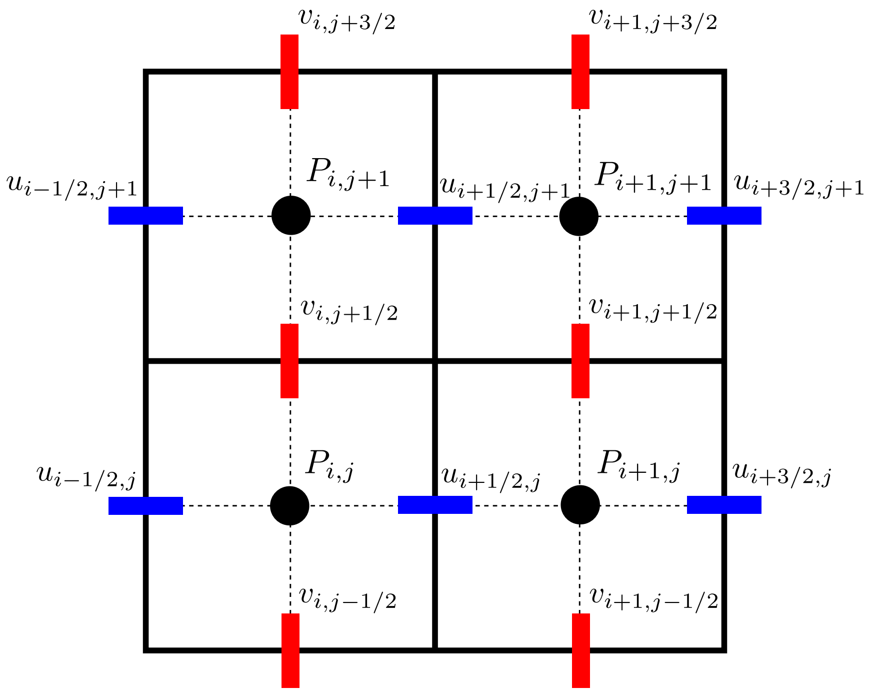


Figure 4.2: A 2D staggered MAC grid with pressure P at the grid-centres represented by circles and velocity components u and v at the cell-edges denoted by the horizontal and vertical marks respectively.

4.2 Standard Methods

A number of methods used in our simulation are standard implementations of well known algorithms and do not warrant in-depth discussion. They are briefly presented here so that it is clear which version we used, along with any modifications that were made for our particular simulator.

Solving for Pressure

The pressure is solved for using the method described by Bridson [7]. Bridson details a specialized modified incomplete Cholesky conjugate gradient method. We instead use the Eigen linear algebra package [25] to solve the sparse pressure matrix for 2D simulations. For 3D simulations a cuSPARSE (part of NVIDIA's CUDA environment) based conjugate gradient is used due to the increased matrix size.

Solving for Viscosity

The viscosity solve modifies the velocities before the pressure solve, but after all external forces have been applied to the fluid as shown in Figure 4.1. We use the method of Carlson et al. [26] which is an implicit solve for viscosity. An important modification from Falt and Roble [27], published a year later, is used to correct the artificial damping of translational motion that was present in the original method. The correction is done by enforcing a Neumann boundary condition $(\nabla\vec{u})\cdot\vec{n} = 0$ at air-fluid interfaces. This boundary condition still does not fix the damping of rotational motion [28]. However, damping of rotational motion is not expected to have a significant effect on a simulation that seeks a static equilibrium along a fluid's interface as is the case in our setting.

Advection and Interpolation

To advect quantities we use the semi-Lagrangian method to solve the advection equation (Equation 3.3) as further detailed by Bridson [7]. As suggested by Bridson, third-order accurate Runge-Kutta is used from Ralston [29]. To interpolate quantities at non-grid locations cubic interpolation is typically used. The interpolant is also taken from Bridson.

4.3 Surface Tracking

The ability to represent and track the position of a liquid's surface is important for fluid simulation. This is especially true when fine detail must be preserved on the surface as it dynamically evolves its position and shape over time. Our simulator labels grid cells according to their contents, so the surface must lie between the sets of fluid and solid/air containing cells. However, labels only indicate whether or not a cell has fluid (empty or full). Surface tracking methods keep a record of where exactly the interface is between these cells. One method, volume of fluid (VOF), assigns a volume fraction between 0 and 1 to each cell to indicate how full it is. Alternatively, the level set method assigns each cell-centre a distance from the interface, as shown in Figure 4.3. Surface tracking therefore allows for a more precise representation of the surface. Here are some of the places that our simulator uses surface tracking.

1. To update material labels which are then used in the pressure solve and viscosity solve.

2. To allow for the selection of the correct permeability coefficients for the magnetic potential solve and subsequent force application.

3. To create the surface mesh that is used for visualization.

4. To calculate surface curvature from, to determine how much surface tension to apply.

Low quality surface tracking will degrade all the above listed steps, especially the last one. Since the Rosensweig instability exhibits a complex surface shape any degradation of the surface quality affects the equilibrium between gravity, surface tension and the magnetic force.

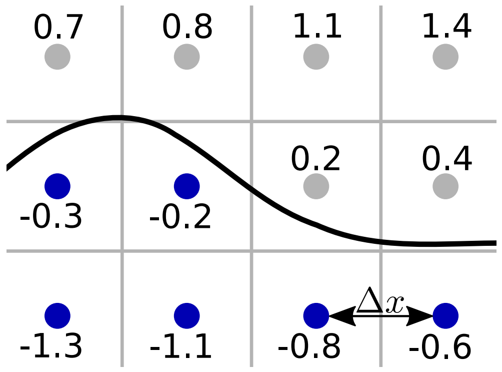


Figure 4.3: A level set stored on a grid for a simple interface. Dark blue circles correspond to cell-centres inside the fluid, while light grey circles are for cell-centres in the air. The level set values stored at each cell-centre are listed adjacent to them. $\Delta x = 1.0$ to allow for intuitive level set values.

4.3.1 Level Set

The level set $\Phi(\vec{x})$ is a scalar field defined as follows

$$\Phi(\vec{x}) \begin{cases} > 0, \text{ when } \vec{x} \text{ is in air} \\ < 0, \text{ when } \vec{x} \text{ is in fluid} \\ = 0, \text{ when } \vec{x} \text{ is on the surface} \end{cases} \tag{4.1}$$

although it is not uncommon to see the signs switched for interface side, since it is an arbitrary convention. The other defining property is the Eikonal equation

$$|\nabla\Phi(\vec{x})| = 1 \tag{4.2}$$

which implies that at the surface the gradient of the level set is also the surface normal.

$$\vec{n} = \nabla\Phi \tag{4.3}$$

These two properties make the level set a signed distance field [30]. At the start of the simulation, with a known geometry, the level set value is analytically calculated at each grid-centre. Initially the level set satisfies Equations 4.1 and 4.2. The level set is then advected in the velocity field during each time step evolving the surface. Advection introduces error and the level set begins to lose its properties, no longer satisfying the Eikonal equation. To correct these errors the level set must be redistanced. First the near-surface level set (where the sign changes) is redistanced.

$$\Phi'_{i,j,k} = sgn(\Phi_{i,j,k})\theta\Delta x \tag{4.4}$$

Here $\theta \in [0,1]$ represents where the surface is located between the two adjacent level set grid values with differing signs.

$$\theta = \frac{\Phi_{i,j,k}}{\Phi_{i,j,k} - \Phi_{i+1,j,k}} \tag{4.5}$$

The remaining cells are redistanced using the Eikonal equation which can be expanded as

$$\left(\frac{\partial\Phi}{\partial x}\right)^2 + \left(\frac{\partial\Phi}{\partial y}\right)^2 + \left(\frac{\partial\Phi}{\partial z}\right)^2 = 1 \tag{4.6}$$

and subsequently discretized as

$$\left(\frac{\Phi_{i+1,j,k} - \Phi_{i,j,k}}{\Delta x}\right)^2 + \left(\frac{\Phi_{i,j+1,k} - \Phi_{i,j,k}}{\Delta x}\right)^2 + \left(\frac{\Phi_{i,j,k+1} - \Phi_{i,j,k}}{\Delta x}\right)^2 = 1 \tag{4.7}$$

where $\Phi_{i+1,j,k}$, $\Phi_{i,j+1,k}$, and $\Phi_{i,j,k+1}$ are the values closest to the surface in this example [7]. Different discretizations are needed depending on which neighbours are closest. Our simulator uses the fast sweeping method, an $O(n)$ algorithm, which applies Equation 4.7 to the non-surface adjacent cells in every possible looping order. The fast sweeping method allows for the smallest cell-to-surface distances to propagate in every direction. An example implementation is provided by Bridson [7].

A Brief Word on Mesh-Based Surface Tracking

In addition to using the level set method for this simulator we also attempted to use mesh-based surface tracking. This is a Lagrangian surface tracking tool meaning that the surface mesh moves with the velocity field. This is in contrast to an Eulerian one, such as level set where the values remain stationary but are updated to reflect surface motion. Individual mesh vertices are initially placed on the interface. They are then advected for each simulation time step evolving the entire mesh and tracking the ferrofluid surface. The key advantage, and one that is attractive for simulating intricate surfaces, is that meshes can preserve a large amount of visual detail [31]. However, to handle fluid merging and separating, as well as keeping the mesh from self-colliding requires additional processing compared to level sets. For example, the El Topo library maintains a collision-free mesh while it is tracking an interface [32]. We integrated this library into our simulator and tested it. While attempting to simulate the Rosensweig instability, we found that peaks separated and merged as they reached equilibrium. It was therefore difficult to maintain a collision free mesh, required for a successful simulation. Additionally, it was difficult to keep the mesh operating at the same resolution as the underlying ferrofluid simulator without using aggressive merging parameters. Once the peak size was below the simulation resolution it was effectively invisible to the velocity field meaning that it would persist unless merged out. The additional processing also resulted in significantly slower runtimes. Therefore it was decided to not further pursue using a mesh for interface tracking.

The discussion above emphasizes the benefits of using level sets for ferrofluid simulation. The level set keeps the fluid matched to the simulation domain resolution since they share the same grid. Any features too small to resolve disappear instead of becoming invisible to the simulation grid and persisting. Merging and separation of fluid bodies occurs automatically without any special treatment. However, the additional detail afforded by using a Lagrangian tracking technique is still desired.

4.3.2 Particle Level Set

The particle level set (PLS) is a hybrid Eulerian and Lagrangian surface tracking method that can combine the benefits of both techniques.

Traditional Particle Level Set

The first such scheme was proposed by Enright et al. in 2002 [33]. Two sets of particles are used. They are seeded close to the interface but on opposite sides. Particles are given either a positive or negative sign, corresponding to interface side. The particles are passively advected in the velocity field using

$$\frac{d\vec{x}_p}{dt} = \vec{u}(\vec{x}_p) \tag{4.8}$$

where $\vec{x}_p$ is the particle position. This is a Lagrangian step. The particles are used as an error detection and subsequent correction mechanism for the level set. The particles are given radii ranging between $0.1\Delta x$ and $0.5\Delta x$. They are randomly placed in the cells surrounding the interface being limited to a particular range, called the bandwidth, such as $3\Delta x$. The particle positions are adjusted such that they sit on randomly selected isocontours of the surface using

$$\vec{x}_{new} = \vec{x}_p + \lambda(\Phi_{iso} - \Phi(\vec{x}_p))\hat{n}(\vec{x}_p) \tag{4.9}$$

where $\hat{n}(\vec{x}_p)$ is the surface normal calculated using the level set at the particle's position. Enright et al. set $\lambda = 1$ unless Equation 4.9 places the updated particle outside the computational domain. Then λ is iteratively halved until the particle lies inside the domain. Equation 4.9 may need to be solved iteratively.

After some simulation time steps take place the surface has moved, updating the level set values and the particle positions. Inconsistencies regarding the location of the interface may exist between the level set and the particles. Particles whose sign does not match the interpolated value from the grid-based level set within a margin of error greater than their respective radii indicate where the level set has degraded. The level set is then redistanced by taking the distance between each level set point and that of the escaped particles. After a number of simulation time steps particles may have strayed from the interface, while remaining on the correct side, requiring that they be reseeded. Existing particles that are near the interface are kept.

Direct Particle Level Set

The surface tracking scheme that we propose to use for our simulator is a modified version of that by Enright et al. [33]. To closely track the interface position we use surface particles that sit directly on it, instead of around it. To determine the sign of the regions adjacent to either side of the interface we use an additional set of signed particles, that sit on their respective side. Both sets of particles are advected in the fluid's velocity field, and are used to reconstruct a grid-based level set approximation of the surface at each time step.

Our surface tracking method operates more closely to a mesh-based one than that of Enright et al. due to placing particles directly on the interface. Therefore, we refer to our surface tracking scheme as a Direct Particle Level Set (DPLS). The idea of using particles directly is a recent idea. Zhao et al. published a PLS method in 2018 that also uses particles directly [34], although with significant differences compared to DPLS. When using DPLS there are two sets of particles, the aforementioned surface particles and positive/negative signed particles. See Figure 4.4 for a visual representation of particle placement. The number of particles for each set is a simulation parameter. An overview of this method is given in Algorithm 1.

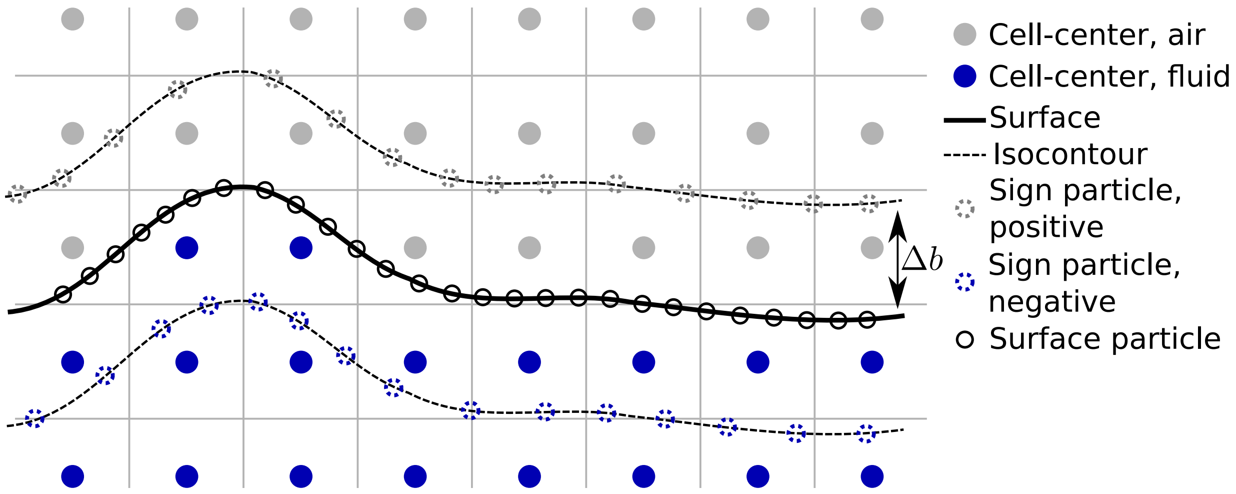


Figure 4.4: Visual depiction of how surface and sign particles might be seeded using the DPLS method. The surface is represented with a solid black line, with air above and fluid below it. Particle density is underrepresented here for clarity. The bandwidth for seeding and the distance of the isocontours are set the same ($\Delta x_{iso} = \Delta b$). Relevant to the massively parallel version, the background grid shows how the particles can be sorted and assigned to individual grid cells for efficient parallel searching.

The seeding algorithm (**reseedParticles**) for each surface particle starts by randomly generating a position for the particle in a grid cell near the interface. The particle is then projected onto the interface using both the level set as well as the gradient (surface normal) of the level set. This requires iteratively solving

$$\vec{x}'_p = \vec{x}_p - \Phi(\vec{x}_p)\nabla\Phi(\vec{x}_p) \tag{4.10}$$

where $-\Phi(\vec{x}_p)\nabla\Phi(\vec{x}_p)$ is a vector from $\vec{x}_p$ to a point on the surface. The seeding error is given by $\Phi(\vec{x}_p)$ since if the particle is exactly on the surface it would be at the zero level set. This process occurs in all cells that are within a certain bandwidth Δb of the surface. There is no point in initially seeding a particle in a cell outside of this bandwidth and trying to iteratively march it towards the interface when it could just be initially placed much closer. The signed particles are seeded along two isocontours, one on each side of the surface. The signed particles' positions are directly calculated from the seeded surface

Algorithm 1: advanceDirectParticleLevelSet()

```
/* Initialize particles on first iteration.                  */
1 reseedParticles();
/* Main time stepping loop.                                  */
2 for $time\ step\ t_i$ do
3     advectParticles();
4     advectLevelSet();
5     redistanceBandwithCells();
6     propagateDistanceOut();
7     correctGridPointSigns();
8     if $i \mod stepsReseed == 0$ then
9         reseedParticles()
```

particles' positions using

$$\vec{x}_{\pm} = \vec{x}_p \pm \Delta x_{iso} \nabla \Phi(\vec{x}_p) \tag{4.11}$$

where $\vec{x}_{\pm}$ indicate positive and negative signed particle positions respectively and Δx_{iso} determines the distance the isocontours sit from the surface. In practise, fewer signed particles than surface particles are needed since the surface details are being captured by the latter. All the particles are advected (**advectParticles**) using Equation 4.8.

On each simulation time step the level set is redistanced from the particles. Using particles directly for redistancing is a key difference from the original method of Enright et al. where only escaped particles are used to correct the level set. The simplest approach to redistancing in our method would be: for every level set grid point, find the closest surface particle, and use its distance to the grid point as the new level set value at that point. This would have $O(n_p n_c)$ complexity where n_p is the number of particles and n_c is the number of grid cells.

However, the only cells that need highly accurate distances are those within the bandwidth of the level set. Instead of calculating new level set values for every cell, one can calculate new values for cells within a neighbourhood of each particle instead (**redistance-BandwidthCells**). Using the particle's position vector, its containing cell can be calculated. If this particle is closer to the cell-centre than any other particle, then this closest particle updates the cell's level set value with the distance between them. A reference to the closest particle is saved. This is now $O(n_p)$ complexity. This process is shown in Algorithm 2. For each cell with a known closest particle, the particle updates its cell's neighbours' level set values with the distance between them and itself, always choosing the

Algorithm 2: redistanceBandwithCells()

```
1 foreach surface particle $\vec{x}_p$ do
2     cellId = getCellId($\vec{x}_p$);
3     cell = getCell(cellId);
4     distance = abs(cell.position - $\vec{x}_p$);
5     if distance < abs(cell.LS) then
6         cell.LS = sgn(cell.LS) * distance;
7         cell.closestParticle = $\vec{x}_p$;
```

lesser of two possible level set values (**propagateDistanceOut**). A similar process is done for the signed particles to assign which side of the surface the cells within the bandwidth region are (**correctGridPointSigns**), always taking the sign of the particle that is closest. For cells outside the bandwidth region the existing sign of the level set is used since the surface is assumed to not move outside of the bandwidth region in a time step. To keep the level set signs updated the level set is also advected on each time step (**advectLevelSet**). Advecting the level set is also done if it is desired to avoid redistancing from particles on each time step to reduce computation time.

Like the method of Enright et al., after a certain number of time steps the particle distribution is likely to no longer provide a sufficiently uniform sampling of the updated surface. At this point all the particles are deleted and reseeded. This would degrade the accuracy in the method of Enright et al. because the particles and interface have been advected separately since the start of the simulation. However, in our method the level set is redistanced directly from the surface particles before reseeding. The particles should lie on the same interface after reseeding, but with a more even distribution. However, there will necessarily be some amount of error introduced due to interpolation of the level set values. The number of time steps for reseeding is a simulation parameter.

4.3.3 Particle Level Set for GPU

Most of the operations of DPLS can be redesigned for a massively parallel computer. We developed an implementation for NVIDIA's GPUs using the CUDA library. Some steps are embarrassingly parallel and do not warrant much discussion. These are typically steps where each thread operates independently of all other threads. A prime example is advection. Each thread represents one particle. Since particles do not interact with each other there is no need for communication between these threads. The input data is the particle's position and the velocity field. The written data is the new particle's position. The velocity field is shared between all threads, but because it is not modified all threads can share read only access to it. The position of each particle can be overwritten with its updated position without issue by its respective thread. Similarly, for advecting the level set each thread can be assigned one value to advect. Lastly, particle seeding is trivial to parallelize. Ahead of time we create an array with a size equal to the maximum number of particles that could be seeded. Each thread attempts to seed a particle. If successful it then writes the particle to the pre-allocated array, indexing it with a unique value (e.g., thread ID). Unfortunately the remaining steps are not trivial to parallelize.

Recall that our method to avoid $O(n_p n_c)$ complexity on level set redistancing was to rephrase the problem in terms of the particles updating the level set values simplifying it to $O(n_p)$ complexity. Attempting to follow the same pattern as for seeding and advection would give each thread a particle to apply Algorithm 2 to. However, there is contention between threads for "Cell.LS" on line 5. Two threads executing simultaneously might both be closer to the cell than any previous particles and therefore try to update the level set value. This creates a race condition and an unknown result for the level set value. It might take on the value assigned by the farther of the two particles, or it might become corrupted, causing redistancing to fail, both undesirable outcomes. Using a coarsely threaded processor (e.g., multicore CPU) mutual exclusion enforced by locks might be practical. With a limited number of threads the probability of two threads processing particles in the same cell would be low. However on the GPU, with thousands of threads, it is likely that many particles in a cell are simultaneously trying to operate on the level set value resulting in performance degradation.

Instead, the original naive cell-based redistancing can be used, albeit with one important modification: sort the stored array of particles using their cell indices as keys. In this implementation the corresponding cells of each particle are calculated on the GPU using one thread per particle. Then a GPU-enabled sort by key algorithm from the CUDA library Thrust is used. Next, the array of particles are binned together by the cell they are closest to as shown in Figure 4.4. Then a thread per cell (within the bandwidth region) can be launched on the GPU. Each thread initially searches only the particles that have been binned for its cell. If no particles are in the thread's cell it will search neighbouring cells up to the preset bandwidth away. This way each particle is searched at most a few times, usually only once, instead of by every grid cell. Each level set value is now being updated by only one thread, removing the previously observed race condition. The complexity is not as ideal as the serial version, while still being $O(n_p)$, it has a larger coefficient, but is far better than the naive implementation's $O(n_p n_c)$ complexity. This same process is repeated to find the closest signed particle per cell within the interface bandwidth, determining the sign of the level set. Level set values are propagated outwards from the bandwidth cells using the CPU since this step does not involve processing of all the particles and is subsequently not performance intensive.

All the steps of this specific PLS implementation that directly operate on or with particles can now take place on the GPU, but with identical surface tracking results. Performance comparisons are given in Chapter 6.

4.4 Extrapolating Velocity

Bridson [7] provides a breadth-first search algorithm for extrapolating velocities from the fluid into the surrounding air. Velocity must be extrapolated since the pressure solve only updates velocities in fluid containing cells. However, velocities in the air adjacent to the fluid surface are used when advecting quantities near the surface. Therefore it is desirable that the air velocities are somewhat valid.

Having valid air velocities is especially important for a ferrofluid simulation where the interface contains high levels of detail that must be maintained as the interface moves. A simple extrapolation from the divergence-free velocities in the fluid adds significant divergence when setting the air velocities. When advection is done for points along the interface, as shown in Figure 4.5, the air velocities are used during the resulting interpolation. This causes non-physical movement of the interface since the velocities are not all divergence-free. This is a critical problem for our particle level set method since particles are being advected near the surface.

To enforce incompressibility of the air velocities a second pressure solve is done after the initial extrapolation. This second pressure solve is identical to the main pressure solve except the liquid interface velocities serve as a Dirichlet boundary condition for the air velocities. No modification of the pressure solve from Bridson is needed to achieve this, other than temporarily labelling the fluid as a solid. If there are multiple regions of air (e.g., the fluid has a pocket of air inside it) each region must be identified and treated separately. The regions are identified by assigning a unique colour to each region of air through a search of connectivity for all air cells.

Note that this procedure is not a replacement for a true two-phase fluid simulation. The cells in the air are essentially treated as an extension of the fluid temporarily to avoid interpolating non-divergence-free velocities for advection inside the fluid. This type of algorithm, an extrapolation followed by a pressure solve, was first proposed by Sussman in 2003 [35] and was popularized in computer graphics by Rasmussen et al. [36] a year later.

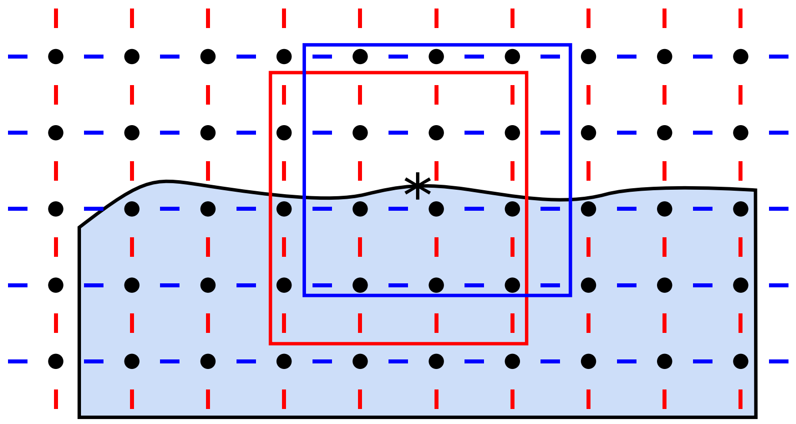


Figure 4.5: The velocity is queried at the indicated point along the surface with cubic interpolation. Vertical and horizontal velocities from both the air and fluid regions are used for interpolation as indicated by the red and blue squares.

4.5 Surface Tension

Surface tension has a pivotal role to play in peak formation for ferrofluids. Recalling Equation 3.21, surface tension is part of the equilibrium of forces that defines the critical magnetization. In Section 3.1 it was mentioned that at the interface Equation 3.7 requires a special case to handle surface tension. Bridson [7] shows that the surface tension results in a pressure jump at the interface equal to

$$P = \sigma\kappa \tag{4.12}$$

where κ is the called the mean curvature of the surface. The interface is unlikely to be positioned at a cell-centre where the pressure is stored. Therefore the surrounding pressure values in the air are set such that the pressure interpolates to $\sigma\kappa$ when $\Phi(\vec{x}_{surface}) = 0$. These air pressures, that normally would be left zero, are called ghost pressures. Applying the surface tension in this explicit manner introduces a new stability restriction on the size of the simulation time step, $\Delta t \leq O(\Delta x^{3/2}\sqrt{\rho/\sigma})$. Improving this restriction is an active area of research and there are semi-implicit approaches [37]. However, some of the methods affect equilibrium shapes and suffer from damping of lower-frequency modes as well as being dimensionally inconsistent [38]. There is not a clear choice for the ideal model of applying surface tension. Our simulation uses the explicit approach.

4.5.1 Determining Curvature

The surface tension force depends on curvature. Therefore any error in the measured curvature will result in erroneous surface tension forces applied to the fluid. Three different methods of curvature measurement are presented. One that is well known (Laplacian-based curvature), one that is an improvement on a leading method (normal-aligned height function for level set), and lastly one we propose specific to surfaces represented using the DPLS method from the previous section (particle based height function for DPLS).

Laplacian-Based Curvature

The mean curvature of the interface can be defined as,

$$\kappa = \nabla \cdot \vec{n} \tag{4.13}$$

where $\vec{n}$ is the surface normal determined from the level set. This implies that $\kappa = \nabla \cdot \nabla\Phi$ gives the mean curvature as a function of the level set [7]. However Osher et al. [30] suggest using

$$\kappa = \nabla \cdot \frac{\nabla\Phi}{|\nabla\Phi|} \tag{4.14}$$

to determine the curvature directly from the surface level set. In practise $|\nabla\Phi| \neq 1$ due to discretization and redistancing errors. Therefore dividing by $|\nabla\Phi|$ rescales the discretized curvature to be correct. Expanding Equation 4.14 yields

$$\kappa = \frac{\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - 2\phi_x\phi_z\phi_{xz} + \phi_z^2\phi_{xx} + \phi_y^2\phi_{zz} - 2\phi_y\phi_z\phi_{yz} + \phi_z^2\phi_{yy}}{|\nabla\phi|^3} \tag{4.15}$$

where subscripts indicate the discrete derivative to take using finite standard differences. The mean curvature can then be determined at all the cell-centres and stored for later use. At specific locations off the grid interpolation can be used to obtain the curvature.

Height Function Based Curvature

Instead of using the level set directly to find curvature, an alternative method is to use the height function. This function calculates the curvature based on the changing height of the interface relative to a common reference. Sussman et al. were the first to use height functions with level sets [39]. Beyond their work, research on height functions has been focused for application to the volume of fluid method (VOF). Afkhami and Bussman introduced their use for 2D and 3D VOF simulations and demonstrated their accuracy compared to a VOF Laplacian method [40] [41]. Improvements have been presented for the VOF version by both Owkes et al. [42] and Popinet [43]. In the present work the height function method is described for level sets. Then the improved version from Owkes et al. is modified and implemented for a level set. Lastly, we propose an implementation of the height function specifically for DPLS.

The height function method relies on the fact that a reference plane can be defined anywhere without affecting the perceived curvature, since the height function uses height derivatives only. In two dimensions, the curvature is measured as

$$\kappa = \frac{h''_y}{\sqrt{1 + {h'_y}^2}} \tag{4.16}$$

where h_y is the height referenced from a horizontal surface $y = h_y(x)$ [38]. If the surface is vertical, then h'_y tends to infinity and h is no longer well defined. In this case a different reference should be used such as $x = h_x(y)$. Sussman et al. [39] and Popinet [38] explain that the height function method is superior to the Laplacian curvature estimate presented in Section 4.5.1 because the height function approximates curvature on the interface as opposed to at cell-centres.

The discretization of the height function as in Equation 4.16 is given using central differences.

$$\kappa_i = \frac{\dfrac{h_{i-1} - 2h_i + h_{i+1}}{[\Delta x]^2}}{\sqrt{1 + \left[\dfrac{h_{i+1} - h_{i-1}}{2\Delta x}\right]^2}} \tag{4.17}$$

The heights can be calculated from the level set by determining in which cells the sign of ϕ changes. The precise location is determined through interpolation with the two surrounding opposite signed cells. This was also done by Sussman et al. [39]. Analytically for a horizontal surface the height is

$$h_i = n\Delta x + \left[\frac{|\Phi(y_n)|}{|\Phi(y_n)| + |\Phi(y_{n+1})|}\right]\Delta x \tag{4.18}$$

where n is the number of cells before the interface was encountered and all Φ are taken along the ith grid column. The height is also calculated using Equation 4.18 for the $i-1$ and $i+1$ columns. Higher order discretizations would require more columns. To implement a discretized height function requires the use of a stencil, so that a height can be determined for each column, as shown in Figure 4.6. In this example, the stencil covers three cells in width, and a variable number of cells in height. Equation 4.18 is used to calculate the height of each column, as shown in red. In practise a stencil seven cells high worked well for our experiments and those by Sussman. Various approaches could be used to select the stencil size automatically, such as by expanding the stencil if a height column does not encounter an interface. The stencil is centred at the requested point of curvature. If one of the heights is not found it is set to zero. This preferentially underestimates the surface curvature rather than causing it to overshoot.

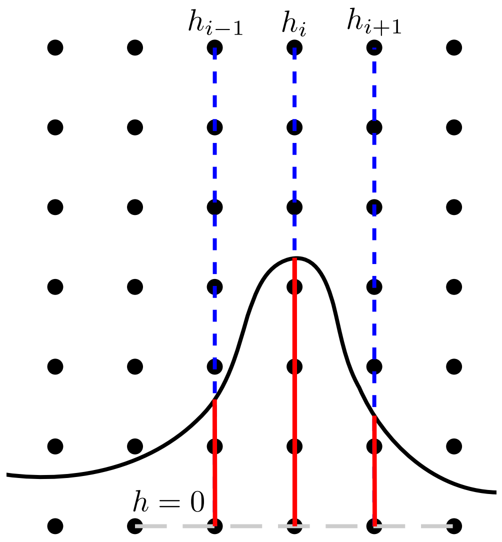


Figure 4.6: The height function stencil applied along a sample interface. The curvature query is where the surface intersects column h_i. Here a 7×3 stencil is used. This size is sufficient to capture the interface heights. The blue dashed line shows each individual height required by Equation 4.17. The red solid lines show the measured heights. The grey horizontal dashed line shows the reference base. Note that a 3×3 stencil would be too small to capture the sample surface geometry.

To select the best reference plane (horizontal or vertical in 2D) the normal determined by the level set is used.

$$\begin{cases} \hat{n} \cdot \{0,1\} > \hat{n} \cdot \{1,0\} \text{ then } y = h_y(x) \\ \hat{n} \cdot \{1,0\} > \hat{n} \cdot \{0,1\} \text{ then } x = h_x(y) \end{cases} \tag{4.19}$$

Naturally, neither choice is optimal for normal vectors that lie between the two cases.

Such a case occurs at a $\pi/4$ angle relative to a grid axis when sampling curvature along the circumference of a circle. An improved method for selecting the reference plane was introduced by Owkes et al. [42]. They suggest using a rotated normal-aligned stencil rather than one fixed to a Cartesian axis. At the point where the curvature is requested the normal is calculated to define a reference line in 2D or a plane in 3D. An additional advantage of a normal aligned stencil is that it results in smaller stencil sizes. In the case of a straight diagonal line at a $\pi/4$ angle to the grid the minimum stencil height remains unchanged. Also, a normal aligned stencil eliminates the need for determining the reference case, thus simplifying the implementation. Owkes et al. used VOF surface tracking, so we present the normal-aligned height function for level set here.

Since we gave the original method in 2D, for completeness we also describe the natural 3D extension. Start with the 3D height function as follows

$$\kappa = \frac{h_{y'y'} + h_{z'z'} + h_{y'y'}h_{z'}^2 + h_{z'z'}h_{y'}^2 - 2h_{y'z'}h_{y'}h_{z'}}{(1 + h_{y'}^2 + h_{z'}^2)^{\frac{3}{2}}} \tag{4.20}$$

where the primed coordinates indicate the appropriate derivatives. All derivatives are taken in a plane with the x-axis aligned to the surface normal. Therefore the plane is defined as

$$\hat{N}_P = \hat{N}(\vec{x}_s) \tag{4.21}$$

$$\vec{x_P} = \vec{x_s} - \frac{n_P}{2}\Delta x \hat{N}(\vec{x_s}) \tag{4.22}$$

where $\hat{N}_P$ is the plane's normal, $\vec{x_s}$ is the point on the surface where curvature is requested, and n_P is the height of the column counted in grid cells. The point $\vec{x}_P$ lies in the plane, and is also the position of the centre height column $\vec{x}(h_{i,j})$. To find the bases of the remaining eight columns requires moving in the reference plane. Positioning of these columns around the central one is arbitrarily set. Using a random vector $\vec{v}$, a vector contained in the plane is found $\vec{v}_P = \hat{N}_P \times \vec{v}$. A perpendicular vector to $\vec{v}_P$ is found using $\vec{v}_P^{\perp} = \hat{N}_P \times \vec{v}_P$. The general position of column $h_{i+k,j+m}$, where $\{k, m \in \mathbb{Z}; -1 \leq k, m \leq 1\}$ can now be defined

$$\vec{x}(h_{i+k,j+m}) = \vec{x}(h_{i,j}) + k\Delta x_P \hat{v}_P + m\Delta x_P \hat{v}_P^{\perp} \tag{4.23}$$

noting that all in-plane vectors have been normalized. Each column is aligned with $\vec{N_P}$ and has a reference allowing for a height to be found. The height calculation is similar to Equation 4.18 for 2D and is given in Algorithm 3.

Step 5 from this algorithm can be replaced using an iterative method for improved accuracy over the existing linear interpolation. Such an interative method was also needed

Algorithm 3: calculateHeightNormal()

```
1 foreach $h_{i+k,j+m}$ do
2     $\vec{x}_h = \vec{x}(h_{i+k,j+m})$
      /* Loop until surface is found with level set sign change.  */
3     while $sgn(\Phi(\vec{x}_h)) == sgn(\Phi(\vec{x}_h + \Delta x \hat{N}_P))$ do
4         $\vec{x}_h = \vec{x}_h + \Delta x \hat{N}_P$
      /* Find the surface fraction.  */
5     $\vec{x}_h = \vec{x}_h + \left[\frac{\Phi(\vec{x}_h)}{\Phi(\vec{x}_h) - \Phi(\vec{x}_h + \Delta x \hat{N}_P)}\right] \Delta x \hat{N}_P$
6     $h_{i+k,j+m} = ||\vec{x}_h - \vec{x}(h_{i+k,j+m})||$
```

to solve Equation 4.10 for the DPLS method. However now the movement of the search $\vec{x}_s$ must be restricted along the height column.

$$\vec{x}_s' = \vec{x}_s - \Phi(\vec{x}_s)(\nabla\Phi(\vec{x}_s) \cdot \hat{N}_P)\hat{N}_P \tag{4.24}$$

The search error is given as the value of $\Phi(\vec{x}_s)$ which indicates how far the search point is away from the zero level set. This improvement can also be used for the regular axis-aligned height function where $\hat{N}_P$ points along the appropriate axis.

The column separation need not match the simulation resolution ($\Delta x_P \neq \Delta x$). Owkes et al. experimented with using $\Delta x_P = \Delta x, 2\Delta x$ or $3\Delta x$. They observed better convergence for larger Δx_P as resolution was increased but worse error at lower resolutions. Choosing $\Delta x_P > \Delta x$ can cause curved features to be ignored despite being at or greater than simulation resolution. We propose a new method for selecting Δx_P. The idea is to choose the largest possible Δx_P without ignoring features at grid resolution. The effect of using variable Δx_P is further analyzed in Section 6.2.3 using DPLS surface tracking.

There is another practical issue with using height functions. Since the columns can sometimes cover a significant portion of the grid to fully capture an interface's curvature it is possible that one column may contain multiple interfaces. This is especially true with features at near-grid resolution. In these scenarios the height that is closest to the position of the requested curvature is used. This is simple to implement by keeping a list of all interface heights found in the column.

Height Functions for DPLS

Using our DPLS surface tracking method introduces the possibility of a new method for measuring surface height. The previous implementation of using the level set to measure height is still compatible. However, the particle positions on the surface can be used directly to determine surface height. The benefit of using particles is a more direct measurement of height, removing the use of the level set except for initial positioning of the stencil. The level set-only approach stores values at cell-centres, and then interpolation is required to determine the height fraction in the surface-containing cell. Instead, using the particle position directly reduces the amount of interpolation used. Algorithm 4 is the DPLS version of Algorithm 3.

Algorithm 4: calculateHeightNormalPLS()

```
1  foreach $h_{i+k,j+m}$ do
2      $\vec{x}_h = \vec{x}(h_{i+k,j+m})$
3      $\varepsilon_{min} = 2n_{max}\Delta x$ /* Set error to maximum value.              */
4      while $\vec{x}_h < n_P \Delta x$ do
5          cellId = getCellId($\vec{x}_h$) /* ID of cell at this position.           */
6          cell = getCell(cellId)
7          foreach $\vec{x}_p$ in cell do
8              $\vec{d} = \vec{x}_p - \vec{x}(h_{i+k,j+m})$ /* Distance from point to column base.   */
9              $h_{candidate} = \vec{d} \cdot \hat{N}_P$/* Project distance along normal.        */
10             $\varepsilon = \sqrt{d^2 - h^2_{candidate}}$/* Distance in reference plane.         */
11             if $\varepsilon_{min} > \varepsilon$ then
12                 $\varepsilon_{min} = \varepsilon$
13                 $h_{i+k,j+m} = h_{candidate}$
14         $\vec{x}_h = \vec{x}_h + \Delta x \hat{N}_P$
```

As before the procedure is repeated for each column in the stencil. The initial error ε_{min} is set to an upper limit, in this case twice the width of the grid. The entire stencil is traversed looking for the closest particle to the column. The process is shown in 2D in Figure 4.7. As described in Section 4.3.2 the particles are sorted into bins corresponding to their closest cell. This is only done for the GPU DPLS algorithm but is a useful preprocessing step to decrease the runtime of this height function. Only particles in the cells covered by the height column need to be searched. The error is the distance from the particle to the column base projected onto the reference plane. The height is this same distance projected along the reference plane normal. The height which corresponds to the closest particle to the column is taken.

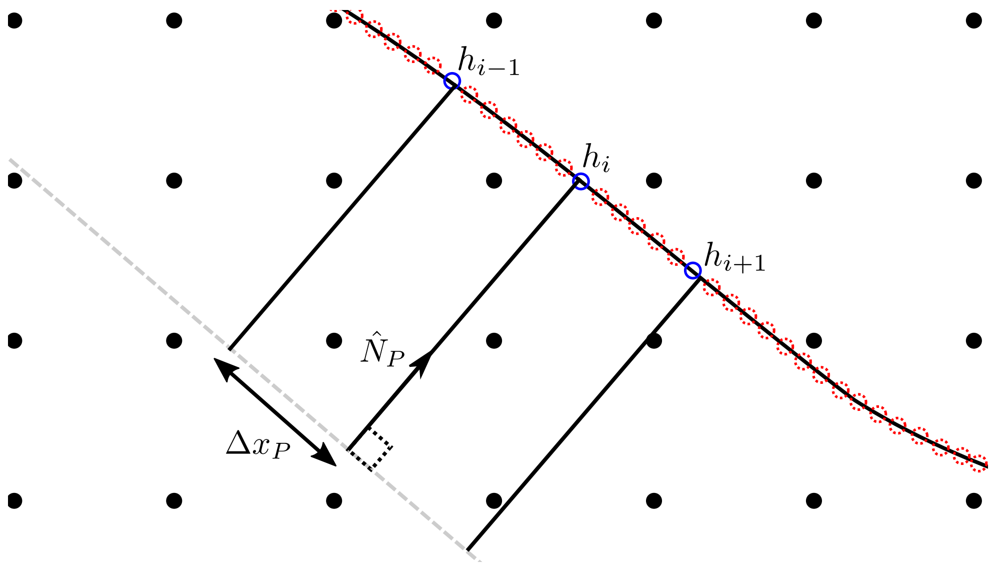


Figure 4.7: The height function stencil is applied to a surface tracked by the DPLS method. The selected particles for each height column are marked with blue circles. The dashed red circles are the remaining surface particles. The dashed grey line is the reference plane. Signed particles are omitted as they are not used to determine curvature.

Some practical considerations may involve saving all heights from particles within a certain tolerance error of the column. This way if multiple interfaces are contained in a column, the height closest to the distance between the point of requested curvature and reference plane can be used. Also if the position along the column $\vec{x}_h$ is on the border of two cells then both cells should be searched to guarantee finding the closest particle. This method is more computationally expensive than the standard level set based height function due to the particle searching.

Chapter 5

Methods for Magnetic Fluids

The Navier-Stokes equation (Equation 3.1) has a term for external forces on the fluid. One common external force is gravity, which is typically a constant vector. Similarly, the magnetic forces that a ferrofluid experiences can also be applied as an external force. However, the magnetic force is not constant, and changes with the magnetization of the fluid. Therefore, a ferrofluid simulator must solve for this magnetic force, which can be found if the magnetic potential is known.

5.1 Solving for Magnetic Potential

Recall that from Section 3.2.2 that Equation 3.14 is $\nabla \cdot (\mu \nabla \Psi) = 0$ where Ψ is the magnetic potential. What is needed are the magnetic forces to apply to the fluid so that it becomes a ferrofluid. However, also recall the relationship between magnetic potential and magnetic force given by Equation 3.13. By knowing the magnetic potential the force can be obtained.

Following the method of Afkhami et al. [17] let us rewrite the magnetic potential as

$$\Psi = \phi + \zeta \tag{5.1}$$

where ϕ is the applied magnetic potential and ζ is the magnetic potential due to the ferrofluid. Splitting the total potential allows us to set boundary conditions for the ferrofluid's potential without considering the contribution due to the applied magnetic potential. The applied magnetic potential is known. For a homogeneous magnetic field the potential is

$$\phi = (G_x x) + (G_y y) + (G_z z) \tag{5.2}$$

where the G coefficients define the strength and direction of the resulting magnetic field after using Equation 3.13. A more interesting magnetic potential to use is

$$\phi = \frac{\vec{m} \cdot \vec{r}}{4\pi|\vec{r}|^3} \tag{5.3}$$

where $\vec{m}$ is the magnetic moment of the magnetic dipole and $\vec{r}$ is the separation vector. There is no inherent constraint with respect to the type of applied magnetic field, as long as it satisfies Gauss's law (Equation 3.8). Note that the corresponding magnetic field of Equation 5.3 does not satisfy Gauss's law in a 2D coordinate system. 2D fluid simulation is commonplace and the physics can be usually adapted by omitting the third basis of the vector space. However, a true 2D simulation of magnetic fields requires a modified magnetic dipole. For example

$$\phi = \frac{\vec{m} \cdot \vec{r}}{4\pi|\vec{r}|^2} \tag{5.4}$$

has a magnetic field that satisfies a 2D Gauss's law and is used as a magnetic source in this work.

In any case, the magnetic potential due to the ferrofluid must be solved for a given applied potential. Substituting Equation 5.1 into Equation 3.14 yields

$$\nabla \cdot (\mu \nabla \zeta) = -\nabla \cdot (\mu \nabla \phi) \tag{5.5}$$

where due to constant μ in free space, Gauss's law (Equation 3.8) and Equation 3.13 the right hand side vanishes in air, and is only non-zero on the interface and inside the fluid.[1] This leaves the problem of boundary conditions along the magnetic simulation domain Ω_M. A Dirichlet condition is undesirable. Conceptually at one point it would merely define the potential's value there, leaving the gradient of the magnetic field unaffected. However, forcing the value along $\partial\Omega_M$ would affect the gradient of the potential. If the domain was infinitely large and far from the ferrofluid using a constant Dirichlet boundary condition would not be a problem since a potential is defined to be constant (e.g., zero) at infinity. However, an infinite domain cannot be handled using this method. A Neumann boundary condition

$$\frac{\partial \zeta}{\partial n} = 0 \tag{5.6}$$

along $\partial\Omega_M$ is preferable instead. Using Equation 5.6 makes the assumption that the boundaries are sufficiently far from the ferrofluid such that the magnetic field there is

[1]This is the case if μ_f varies, for example as in Equation 3.19. If μ_f is constant in the fluid then the right hand side also vanishes, except at the interface.

approximately uniform. Such an approximation is a possible area of error. One way to mitigate this error is to define the magnetic simulation domain to be larger than the fluid simulation domain. The magnetic potential due to the ferrofluid is increasingly uniform farther away from the fluid. As a result, placing the magnetic simulation domain boundaries farther from the fluid better meets the assumption made by Equation 5.6.

5.1.1 Eliminating the Null Space

Notice that the only type of boundary conditions defined are Neumann. Previous simulations had symmetry conditions [9] or knew approximately the geometry of their ferrofluid and could use a Dirichlet condition [17] . Unfortunately, a general simulation is unable to impose additional boundary conditions since the geometry of the fluid body is unknown a priori. Equation 5.5 is therefore a **pure Neumann** varying coefficient Poisson problem. Fortunately this form of equation has been considered in scientific computing, and a solution for solving it is taken from Yoon et al. [44].

The varying coefficient μ is ignored for now as Yoon et al. present their solution for the more general Poisson equation.

$$\begin{aligned} \Delta u &= f \text{ in } \Omega \\ \frac{\partial u}{\partial n} &= g \text{ on } \partial\Omega \end{aligned} \tag{5.7}$$

The solution exists if the following compatibility condition is satisfied.

$$\int_{\Omega} f dx + \int_{\partial\Omega} g ds = 0 \tag{5.8}$$

However, the solution is not necessarily unique. A constant can be added to u and still satisfy Equation 5.7. The additional condition

$$\begin{gathered} \int_{\Omega} u dx = 0 \text{ or} \\ u = 0 \text{ for a point } q \in \Omega \end{gathered} \tag{5.9}$$

guarantees uniqueness of the solution. There is no such physically mandated condition for Equation 3.14. Fixing the value at one point would not violate any physical law (only the gradient of the potential is being used), but Yoon et al. show that fixing a value introduces a sink or source at this point to the solution.

They prove that the correct method is to first guarantee existence by projecting b^h onto the range space $R(A^h)$ where

$$A^h u^h = b^h \tag{5.10}$$

is the linear system corresponding to Equation 5.7. After projection b^h can be represented as a combination of the columns in A^h. Projection simply requires subtracting the mean of b^h from each element within b^h. Bridson shows that the same step must be done for the pressure solve [7]. Next, to guarantee uniqueness it is necessary to ensure that the solution u^h is chosen from ker $(A^h)^\perp$ where $\ker(A^h) = \{v^h \in A^h | A^h v^h = 0\}$. The same procedure is done as for guaranteeing the existence of the solution: subtract the mean of the elements from the residual and search vectors of conjugate gradient on each iteration. The modified conjugate gradient algorithm of Yoon et al. is given in Algorithm 5. The added steps are listed in bold. The parameter x^n is the approximate value for u^h in Equation 5.10 after n steps, r^n is the residual and p^n is the search direction. The inner product is defined as $\langle \alpha, \beta \rangle_h = \sum_{i,j} \alpha_{ij} \beta_{ij} h^2$. The parameter h is the simulation resolution Δx.

Algorithm 5: Modified Conjugate Gradient

```
1 for $n$ = $1, 2, \ldots,$ do
2     $x^{n+1} = x^n + \frac{\langle r^n, r^n \rangle_h}{\langle p^n, A^h p^n \rangle_h} p^n$
3     $r^{n+1} = r^n - \frac{\langle r^n, r^n \rangle_h}{\langle p^n, A^h p^n \rangle_h} A^h p^n$
4     $\mathbf{r^{n+1} = r^{n+1} - \frac{\langle r^{n+1}, 1^h \rangle_h}{\langle 1^h, 1^h \rangle_h} 1^h}$
5     $p^{n+1} = r^{n+1} + \frac{\langle r^{n+1}, r^{n+1} \rangle_h}{\langle r^n, r^n \rangle_h} p^n$
6     $\mathbf{p^{n+1} = p^{n+1} - \frac{\langle 1^h, p^{n+1} \rangle_h}{\langle 1^h, 1^h \rangle_h} 1^h}$
```

Following these steps Eigen's conjugate gradient solver and NVIDIA's sample CUDA conjugate gradient solver were both modified to support the solving of pure Neumann Poisson problems.

5.1.2 The Interface Condition

Now that the issues due to boundary conditions have been resolved, consider what happens at the interface due to the varying coefficient μ. The interface conditions are

$$\begin{aligned} \Psi_a - \Psi_f &= 0 \\ (\mu_a \nabla \Psi_a - \mu_f \nabla \Psi_f)\hat{n} &= 0 \end{aligned} \tag{5.11}$$

where the subscripts a and f denote components in air and ferrofluid respectively [20].

These conditions allow this particular problem to benefit from a method published by Kang et al. without further work [45]. Their method is specialized for the varying coefficient Poisson equation where the coefficient (μ here) may be discontinuous. Their method can also handle cases when the solution itself is discontinuous. This numerical method does not suffer from numerical smearing and is first order accurate. Kang et al. only modify the right hand side of the Poisson equation. Since both interface conditions for this specific problem are zero, these proposed right hand side terms become zero, returning the original equation.

The last part of the method by Kang et al. involves interpolation when accessing the varying coefficient at half points between cell-centres. They propose using

$$\mu_{k+1/2} = \frac{\mu_{k+1}\mu_k(|\Phi^k| + |\Phi^{k+1}|)}{\mu_{k+1}|\Phi^k| + \mu_k|\Phi^{k+1}|} \tag{5.12}$$

where the specific variables have been substituted in for this problem [2]. However, they mention no strict requirement to use this specific interpolation method. Instead, we use the following equation

$$\mu_{interpolated} = \mu_f + (\mu_a - \mu_f)\left(\frac{tanh(\alpha\Phi) + 1}{2}\right) \tag{5.13}$$

taken from [46]. Contrasting with Equation 5.12, Equation 5.13 gives the average of the values when the interface is exactly in between two cell-centres. Equation 5.13 can also be tuned using the α parameter. Lastly, Equation 5.13 naturally returns to either μ_a or μ_f outside of the interpolation region and can be used anywhere the level set is defined. The downside is that Equation 5.13 requires an additional step of interpolation, however for the common case of $\mu_{k+1/2}$, the result is just the average of the two surrounding level

[2]This is from Equation 55 in [45], which has a suspected typo where a β^+ was replaced by a Φ^+ in the denominator. The more likely intended equation is used here.

set points. Regardless of method, smooth interpolation for μ has the benefit that it avoids suddenly switching from μ_a to μ_f when the fluid first occupies the point in space where $\mu_{k+1/2}$ is located. Visually this produces an interface oscillation around that point since an equilibrium of forces can not be achieved. For this reason, anytime μ is required near the interface the level set based interpolation from Equation 5.13 is used.

5.1.3 Discretization

A Cartesian discretization is needed of Equation 3.14. The original method of Afkhami et al. shows a derivation for axisymmetric coordinates [17], which does not meet the needs for a general simulation. The Cartesian grid discretization used here is

$$\begin{aligned}\nabla \cdot (\mu \nabla \zeta)_{i,j,k} = & \frac{\mu_{i+1/2,j,k}\left(\frac{\partial \zeta}{\partial x}\right)_{i+1/2,j,k} - \mu_{i-1/2,j,k}\left(\frac{\partial \zeta}{\partial x}\right)_{i-1/2,j,k}}{\Delta x} \\ & + \frac{\mu_{i,j+1/2,k}\left(\frac{\partial \zeta}{\partial y}\right)_{i,j+1/2,k} - \mu_{i,j-1/2,k}\left(\frac{\partial \zeta}{\partial y}\right)_{i,j-1/2,k}}{\Delta x} \\ & + \frac{\mu_{i,j,k+1/2}\left(\frac{\partial \zeta}{\partial z}\right)_{i,j,k+1/2} - \mu_{i,j,k-1/2}\left(\frac{\partial \zeta}{\partial z}\right)_{i,j,k-1/2}}{\Delta x}\end{aligned} \tag{5.14}$$

where

$$\left(\frac{\partial \zeta}{\partial x}\right)_{i+1/2,j,k} = \frac{\zeta_{i+1,j,k} - \zeta_{i,j,k}}{\Delta x} \tag{5.15}$$

and similarly for the remaining five partial derivatives. The discretization simplifies to a form similar to the standard Laplacian one

$$\begin{aligned}\nabla \cdot (\mu \nabla \zeta)_{i,j,k} = \frac{-1}{\Delta x^2}\big[& \mu_{i+1/2,j,k}\zeta_{i+1,j,k} + \mu_{i-1/2,j,k}\zeta_{i-1,j,k} \\ & + \mu_{i,j+1/2,k}\zeta_{i,j+1,k} + \mu_{i,j-1/2,k}\zeta_{i,j-1,k} \\ & + \mu_{i,j,k+1/2}\zeta_{i,j,k+1} + \mu_{i,j,k-1/2}\zeta_{i,j,k-1} \\ -(\mu_{i+1/2,j,k} + \mu_{i-1/2,j,k} & + \mu_{i,j+1/2,k} + \mu_{i,j-1/2,k} + \mu_{i,j,k+1/2} + \mu_{i,j,k-1/2})\zeta_{i,j,k}\big]\end{aligned} \tag{5.16}$$

except the coefficients are in terms of μ instead of whole numbers. The right hand side (RHS) is set as

$$RHS = \begin{cases} 0 & \text{in air} \\ \nabla \cdot (\mu \nabla \phi) & \text{in fluid or at interface} \end{cases} \tag{5.17}$$

recalling that ϕ is the known potential of the applied field. While ϕ is known analytically, μ is a discrete quantity, so the same discretization as ζ is used for the RHS.

The final system of linear equations is solved using Eigen's conjugate gradient method with diagonal preconditioner. On CUDA-enabled hardware, the sample CG program provided with the CUDA library is used. Along with the modifications for solving a pure Neumann boundary condition problem, the CUDA CG program was modified to include a diagonal preconditioner.

5.2 Applying the Magnetic Force

With the magnetic potential of the ferrofluid solved, the forces on the fluid can be applied. First, remember that the solve is just for the ferrofluid potential ζ since the potential of the applied magnetic field ϕ is known. We can then use Equation 5.1 to get the total potential Ψ when calculating the force. Also recall Equation 3.13 which calculates the magnetic field from the potential. The velocity update due to magnetic forces is given by

$$\frac{\partial \vec{u}}{\partial t} = \frac{1}{\rho}\nabla \cdot \tau \tag{5.18}$$

where τ is the magnetic stress tensor [17]. The magnetic stress tensor is a rank two symmetric tensor. A stress tensor such as this contains the distribution of stress components for a 3D cubic volume. One dimension of the tensor sets the side the force is applied on. The other dimension sets the direction of that force.

$$\begin{aligned} \tau_{ij} &= \mu \left(H_i H_j - \frac{|\vec{H}|^2 \delta_{ij}}{2} \right) \\ \delta_{ij} &= \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \end{aligned} \tag{5.19}$$

This definition for the magnetic stress tensor is given by Afkhami et al. [18]. Stierstadt and Liu [47] show how to derive the magnetic stress tensor under a number of different cases. The case for ferrofluids is a particularly simple example since there are no electric field terms. The only difference from the derivation by Stierstadt and Liu is that μ is set to its respective material (either μ_f or μ_a here).

Next the divergence of the tensor is taken.

$$\begin{aligned}\nabla \cdot \tau = &\left[\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z}\right]\hat{x} \\ &+ \left[\frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z}\right]\hat{y} \\ &+ \left[\frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}\right]\hat{z}\end{aligned} \tag{5.20}$$

The result will be referred to as a vector (T_x, T_y, T_z) for future convenience. Each component of this vector is then applied to its partner velocity component, e.g., $u_x' = u_x + (\frac{\Delta t}{\rho})T_x$. Therefore, each of the partial derivatives in Equation 5.20 needs to be discretized so that their central differences are centred at the velocity update locations on the MAC grid.

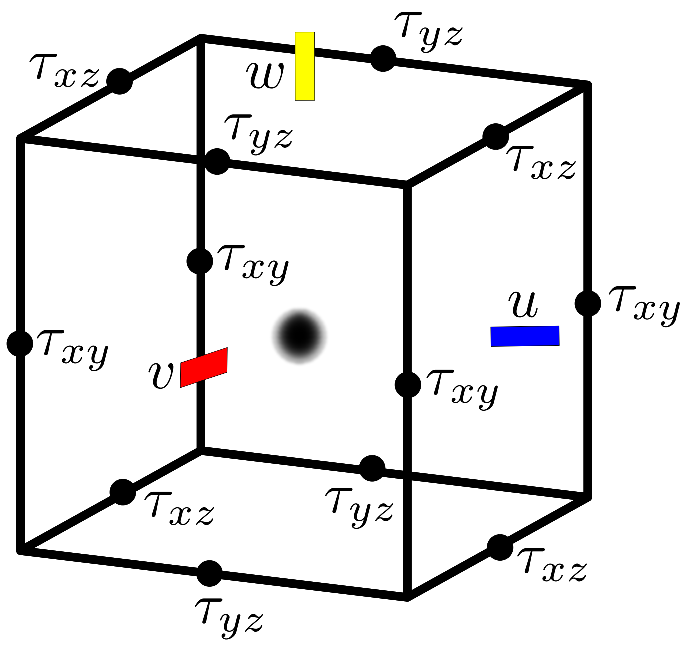


Figure 5.1: The locations of each tensor component are shown on a grid cell. The point at the centre is the cell-centre where the terms τ_{xx}, τ_{yy} and τ_{zz} are stored, as well as other cell-centred data such as pressure. The velocity vector components u, v and w are located at the red, blue and yellow boxes respectively. Indices are omitted for visual clarity.

The most straightforward terms are τ_{xx}, τ_{yy} and τ_{zz} since they only contribute to each velocity component once, and in the direction of their derivatives. See Figure 5.1 to see where each component of the tensor is located.

As an example, the central difference of

$$\left[\frac{\partial \tau_{xy}}{\partial x}\right]_{i+\frac{1}{2},j,k} = \frac{\tau_{xy(i+\frac{1}{2},j+\frac{1}{2},k)} - \tau_{xy(i+\frac{1}{2},j-\frac{1}{2},k)}}{\Delta x} \tag{5.21}$$

gives the velocity update at grid index $(i+\frac{1}{2},j,k)$ which is the location of $u_{i+\frac{1}{2},j,k}$. The tensor component is then located ideally for updating $v_{i,j+\frac{1}{2},k}$ with another central difference. This procedure is done for every tensor component to apply the ferrofluid forces as a velocity update on the grid.

Chapter 6

Results

Before presenting the results of the ferrofluid simulation the methodology used needs to be verified. The Rosensweig instability relies on the delicate balance between surface tension, gravity and magnetic force. Poor results in one of these areas can prevent this phenomena from occurring. Previously discussed in Section 4.3 was the importance of an accurate surface tracking method, since it impacts multiple parts of the simulator, including curvature measurement. Therefore, the performance of our DPLS surface tracking method is tested against a standard level set. Then we compare different curvature measurement techniques. Using the best performing one, surface tension is qualitatively verified. Some basic fluid simulations are then shown. Finally, having tested the fluid simulator, the full ferrofluid simulator is used to reproduce a couple ferrofluid phenomena, including the Rosensweig instability.

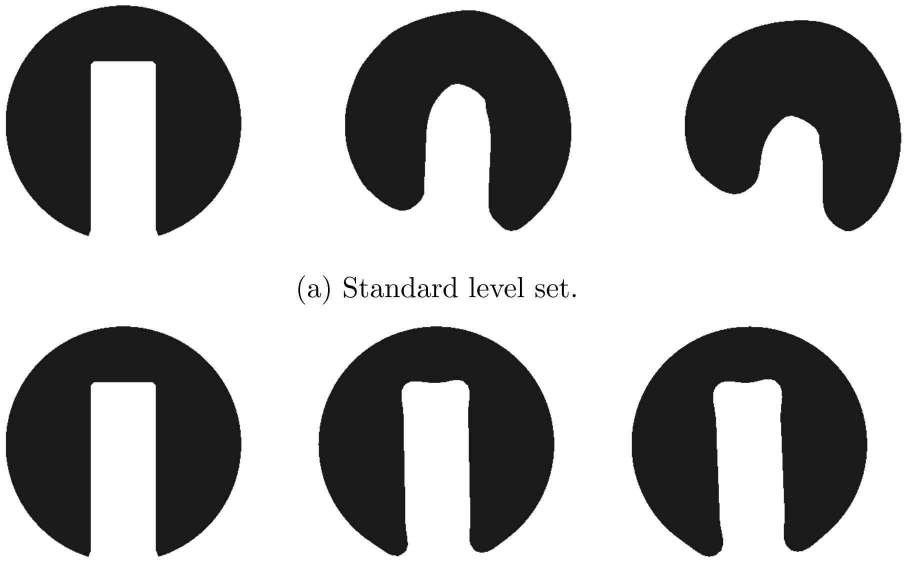
6.1 Surface Tracking Comparison

A standard test of surface tracking methods is to advect a rigid body through a velocity field. Ideally, the rigid body arrives unchanged. In reality, depending on the accuracy of the method used, there will be some deformation from the original shape. A popular setup is to use Zalesak's disk in a constant vorticity velocity field [33]. The velocity field is defined by

$$\begin{aligned} u &= \left(\frac{\pi}{314}\right)\left(\frac{n_y}{2} - y\right) \\ v &= \left(\frac{\pi}{314}\right)\left(x - \frac{n_x}{2}\right) \end{aligned} \tag{6.1}$$

where n_x and n_y are the domain dimensions. Using this definition the centre of rotation matches the centre of the domain. Zalesak's disk is a circle with a radially-aligned rectangular slot. The disk is offset from the centre of the domain. The period of rotation is 628 time units.

We used a 120×120 grid for this experiment. The disk has a 20 unit radius, and a 10 unit wide slot that cuts 30 units deep into the circle. The disk is initially centred at (60, 90). The disk is rotated two times around the domain centre for both standard level set and DPLS. The disk is shown in Figure 6.1 at the experiment start, after one rotation and after two rotations.



(a) Standard level set.

(b) Direct particle level set (DPLS).

Figure 6.1: Frames of a rotating Zalesak's disk are shown. The times for each frame are 0, 625 and 1255 time units respectively, totalling two periods of rotation.

To ensure a fair comparison both surface tracking methods redistance their level set after each time step. DPLS has the additional step of reseeding its interface, which is done every 10 time steps. The disk experiences some deformation when rotated with both methods. However, when rotated using DPLS the deformation is slight compared to using standard level set. Numerically, the disk rotated with standard level set loses significant mass, which can be quantified with area. There is a 9.75% reduction in area after two rotations, while using DPLS results in only a 0.25% reduction. Clearly DPLS offers better surface tracking accuracy. The positive effect of this will later be shown when measuring surface curvatures.

6.2 Surface Curvature Measurement

The only variable, besides the scaling coefficient, that determines surface tension for this simulation is surface curvature. Therefore the accuracy and robustness of surface tension is heavily reliant on the curvature method. Testing curvature measurement requires applying the method to an interface with theoretically known curvature. As in prior work [42] the circle will be used where the curvature κ_c is simply $1/R$ where R is the radius. The level set is then defined

$$\Phi(\vec{x}) = |\vec{x} - \vec{x}_c| - R \tag{6.2}$$

where $\vec{x}_c$ is the centre of the circle. The error is now defined between the expected curvature κ_c and the measured curvature κ.

$$\varepsilon = |\kappa_c - \kappa| \tag{6.3}$$

Each method was tested using the following procedure. Multiple trials were performed to prevent biases due to grid position and alignment among the methods. Every method was run on each trials' data so that direct per-trial comparison of data is possible. Each trial consisted of the following steps. First a random circle position $\vec{x}_c$ was generated inside the simulation domain. Then a radius was also randomly chosen. Along the boundary of the circle the curvature was requested using each method, uniformly sampling the entire circumference. An example of a trial is given in Figure 6.2. At least 1000 sample points were used to capture the methods' errors at all circle angles from 0 to 2π. Using sampling that is too coarse risks missing sudden jumps in error at particular angles for each method. All other parameters including resolution, and scale remained fixed for each batch of trials unless otherwise noted.

Both L2 and L-inf error metrics are used. L-inf is useful when an otherwise accurate method has comparatively isolated cases where large errors can occur. Large errors in curvature result in incorrect surface tension forces propagating into the simulation. L2 gives a better estimate of the total error for all trials. The error is presented in two ways. First, as a function of the angle (position) along the circle's circumference. This identifies how a method's accuracy varies with grid orientation. The two error metrics as a function of angle θ are defined as

$$\mathrm{L2}(\theta_j) = \frac{1}{N_s}\left[\sum_{i=1}^{N_s}\left[\frac{\kappa_i(\theta_j) - \kappa_{c,i}(\theta_j)}{\kappa_{c,i}(\theta_j)}\right]^2\right]^{\frac{1}{2}} \tag{6.4}$$

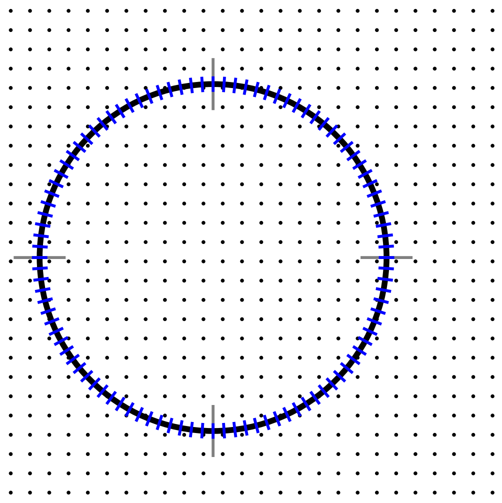


Figure 6.2: The curvature of a randomly generated circle is uniformly sampled 100 times along its complete circumference. These points are denoted using short blue lines. Longer grey lines indicate the angles 0, $\pi/2$, π and $3\pi/2$. Denser sampling is typically used.

$$\text{L-inf}(\theta_j) = \max_{i=1\ldots N_s} \left| \frac{\kappa_i(\theta_j) - \kappa_{c,i}(\theta_j)}{\kappa_{c,i}(\theta_j)} \right| \tag{6.5}$$

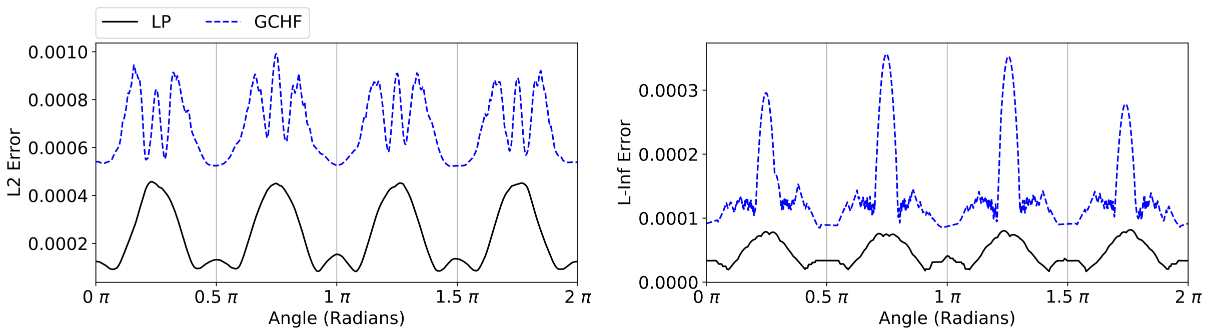
where N_s is the number of trials. Next the error for all trials and angles can be presented as a single value per method.

$$\text{L2} = \frac{1}{N_s} \left[\sum_{i=1}^{N_s} \frac{\left[\sum_{j=1}^{N_a} (\kappa_i(\theta_j) - \kappa_{c,i}(\theta_j))^2\right]^{\frac{1}{2}}}{\left[\sum_{j=1}^{N_a} \kappa_{c,i}(\theta_j)^2\right]^{\frac{1}{2}}} \right] \tag{6.6}$$

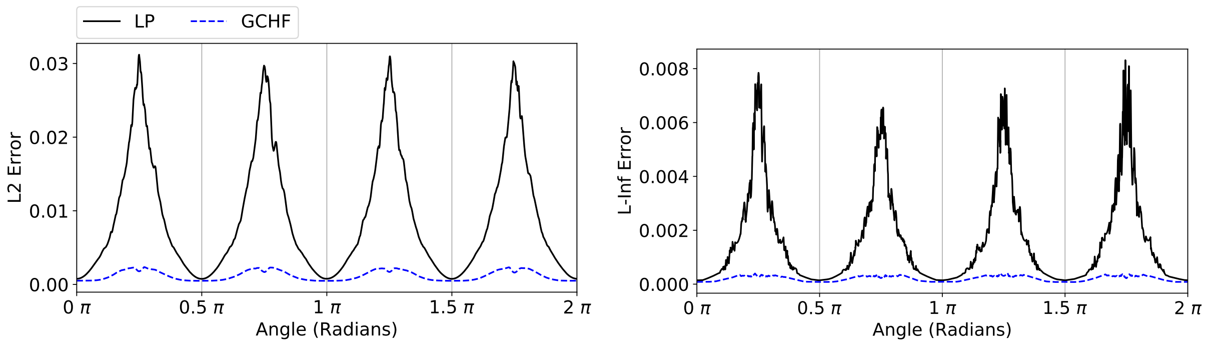
$$\text{L-inf} = \max_{i=1\ldots N_s} \left[\max_{j=1\ldots N_a} \left| \frac{\kappa_i(\theta_j) - \kappa_{c,i}(\theta_j)}{\kappa_{c,i}(\theta_j)} \right| \right] \tag{6.7}$$

Here N_a is the number of angles, which is the number of points sampled. All errors are taken relative to the curvature to avoid high curvatures from dominating the results. Equations 6.6 and 6.7 are similar to those used by Owkes et al. [42].

The level set was redistanced in advance of each trial using the method given in Section 4.3.1, which solves the discrete Eikonal equation (Equation 4.7). Visually the initialized circle remains unchanged, however redistancing introduces error into the level set. A simulation uses redistancing constantly to maintain the level set properties. Therefore a suitable curvature method for simulation must give reliable curvatures on redistanced data. A motivating example shown in Figure 6.3 compares the Laplacian curvature (Section 4.5.1) to a well performing height function method, grid-centred height function (GCHF), before and after the redistancing step.



(a) Before level set redistancing.



(b) After level set redistancing.

Figure 6.3: The effect of redistancing on curvature measurement accuracy between Laplacian (LP) and a leading height function method (GCHF). Simulations are performed on a 64×64 unit grid with circle radii distributed between 3 and 21 units.

The standard Laplacian method shows the best results for a level set initialized directly for a circle. Both L2 and L-inf total errors significantly increase by about two orders of magnitude for the Laplacian method after the level set has been redistanced. See Table 6.1 for exact errors. The height function method is far more robust to the errors introduced by redistancing. The relationship of the error to angle is different before and after redistancing, but the total error change is minor in comparison. The L2 error approximately doubles while the L-inf error shows even less increase. The robustness of the height function method could be, as explained by Sussman [39] and Popinet [38], due to the fact it samples curvature directly at the interface.

Method	No Redistancing		With Redistancing	
	L2	L-inf	L2	L-inf
Laplacian	0.00372	0.0286	0.223	3.03
GCHF	0.012	0.124	0.0257	0.151

Table 6.1: Total L2 and L-inf errors for the grid-aligned height function and Laplacian curvature measurement methods before and after level set redistancing.

Another common trend among all data is shown in Figure 6.3. The curvatures tend to be significantly more accurate at grid aligned locations as indicated by grey lines in Figure 6.2. A similar trend can be seen in the work by Owkes et. al. [42]. This trend is periodic with the angle of the requested curvature relative to the grid axes. Therefore future errors that are shown as a function of angle are limited to one period only (the circumference between angles 0 and $\pi/2$).

6.2.1 Curvature Method Comparison for Levelset

Clearly from the previous results for redistanced data the GCHF method is superior to the Laplacian. However, there are a number of different competing implementations of the height function. The variations and abbreviations used are shown in Table 6.2. The two main groups are the axis-aligned and normal-aligned height functions. They also have grid-centred (GC) variants. Instead of positioning the height stencil exactly at the position of requested curvature, the curvature is instead calculated at all grid-centres beforehand. Then at runtime the values are interpolated to the specific locations on the circles. To fully observe the performance differences randomized circle tests are used as before, however resolution becomes another variable. Specifically, the performance for high curvature circles, such that they have a low resolution on a consistent grid, can serve as a stress test. These low resolution high curvature tests may reveal previously unseen differences between the methods. Therefore all tests are done for circles with radii between 2 and 5 grid cells, and then again for radii between 5 and 43 cells. Before analyzing their comparative performance there are some individual and shared optimizations that need to be tested.

Name	Abbreviation	Description
Height Function	HF	Axis aligned height function centred at the position of requested curvature.
Normal Aligned Height Function	NAHF	Surface normal-aligned height function centred at the position of the requested curvature.
Grid Centred Height Function	GCHF	Axis aligned height function that only operates on cell-centred locations. Curvature at off centre positions is interpolated from saved node values.
Grid Centred Normal Aligned Height Function	GCNAHF	Surface normal-aligned height function that only operates on cell-centred locations. Curvature at off centre positions is interpolated from saved cell values.

Table 6.2: The varying height function methods tested in this work.

Iterative Search

The first such optimization, as explained in Section 4.5, is to use an iterative method for determining the fractional height in a partially filled cell of fluid. The HF and NAHF methods are selected for this test. The first set of results are shown in Table 6.3. The goal here is to first optimize the methods so that the best versions can be compared in the next section. Direct comparison is saved until then.

Method	Interpolation		Iterative	
	L2	L-inf	L2	L-inf
HF	0.158	2.52	0.143	2.40
NAHF	0.159	2.54	0.145	2.42

Table 6.3: Total L2 and L-inf errors for HF and NAHF methods comparing the use of the iterative method to the original interpolation method for calculating column heights. Circles had radii between 5 and 43 units.

A minor decrease in error using the iterative search for both methods is observed. For small geometries (radii between 2 and 5 cells) a larger improvement using the iterative search was observed. This is an important consideration for ferrofluids due to the potential small scale of the Rosensweig instability. The high curvature data is presented in Table 6.4.

Method	Interpolation		Iterative	
	L2	L-inf	L2	L-inf
HF	0.149	1.94	0.132	1.94
NAHF	0.162	0.685	0.113	0.500

Table 6.4: Total L2 and L-inf errors for HF and NAHF methods for high curvature data only (radii of circles between 2 and 5 units).

While the improvements were similar between HF and NAHF methods for the low curvature tests, now the NAHF method has significantly better error using the iterative method (see Table 6.4). The HF method generally improved although to a much smaller degree than NAHF did. As will be further explored in the next section, HF does not perform well on small geometries due to poor alignment of its stencil with the interface. This is especially true when non-axis-aligned such as near $\pi/4$. This error makes the benefits of iterative search less relevant for HF.

Reviewing the total L2 and L-inf errors the overall error decreased for every dataset and height function variant using the iterative method. The runtime cost compared to the rest of the simulation is minor so for ease of implementation the iterative method is used, unless otherwise stated, for all height functions, although the benefit is mostly for low resolution high curvature geometry. This is likely due to interpolation losing accuracy as the geometry is being represented with fewer grid cells.

Height Function Comparison

Naturally the question becomes which height function is superior. HF and NAHF are tested along with their grid-centred (GC) variants. They all use the iterative method except for GCHF. Note that for the GCHF the iterative method performs similarly to straight interpolation. Since the stencil is always grid-centred interpolating the fractional height only requires linear instead of bilinear interpolation. The GCNAHF method, while also calculating curvature at grid-centred locations, does benefit from using the iterative method since its columns are not grid aligned. All the methods are tested as before, on randomly generated circles. The first test is for low curvatures (radii between 5 and 43 cells).

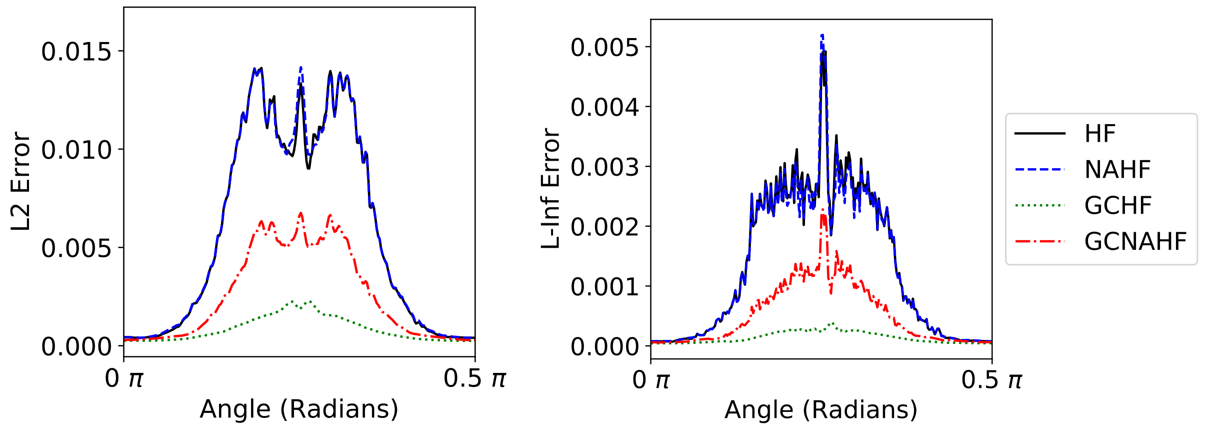


Figure 6.4: The errors of all height functions for low curvatures. All heights are measured using the iterative method except for GCHF. These simulations were performed with radii between 5 and 43 units.

Method	L2	L-inf
HF	0.141	2.00
NAHF	0.144	2.11
GCHF	0.0210	0.162
GCNAHF	0.0634	0.929

Table 6.5: Total L2 and L-inf errors for HF, NAHF, GCHF and GCNAHF methods for circle radii between 5 and 43 units.

The total errors recorded in Table 6.5 show that the GCHF method performs the best over all tests followed by GCNAHF and a near tie between HF and NAHF. The GC variants outperform their counterparts most likely due to the decrease in interpolation required. The similar performance between HF and NAHF further supports this reasoning. When the curvature is well defined, such as for circles with a radius of multiple cells, the interpolation errors of HF and NAHF outweigh any other improvements. Although they use the iterative method, interpolation is still used to find the zero level set. Not only are their total errors close or identical, the dependency of the error due to angle, as seen in Figure 6.4, is also identical. However, there is a difference when using normal alignment for the GC family of methods.

GCHF strongly benefits from reduced use of interpolation compared to GCNAHF for finding column heights. Since GCHF uses columns aligned with the grid, interpolation is always linear between two points. GCNAHF also places its reference base at a grid-centre but because it is normal-aligned the columns are not grid aligned requiring bilinear interpolation of the level set. A simple use of error propagation shows that linear interpolation has less error than bilinear. Therefore GCHF will benefit from less error due to using only linear interpolation. However, at near axis-aligned angles GCNAHF essentially uses linear interpolation giving a performance improvement over NAHF. The performance improvement of both GC methods is also partially due to the averaging effect of storing curvatures at grid-centres. Since the circular geometry has consistent curvature all the stored curvature values at grid-centres should be the same. Interpolation for off grid points essentially averages these stored values. Therefore when using a synthetic test with constant curvature the GC variants will have an advantage.

Method	L2	L-inf
HF	0.142	1.96
NAHF	0.116	0.512
GCHF	0.0906	0.618
GCNAHF	0.0812	0.450

Table 6.6: Total L2 and L-inf errors for HF, NAHF, GCHF and GCNAHF methods but now for circle radii between 2 and 5 units.

For curvatures of small geometries, in this case circles with radii between 2 and 5 grid cells, GCHF is no longer the best method. Referring to Table 6.6, GCNAHF has the best overall performance. Picking GCHF or NAHF as the next best performing depends on the priority given to L2 versus L-inf error while HF is the worst performing. The HF/GCHF stencil, while three cells wide like the NAHF/GCNAHF height functions, will poorly sample the surface since the stencil is not well positioned over it. Normal alignment allows the stencil to rotate to better fit on small geometries. When the stencil does not fit the geometry, as shown in Figure 6.5, a column might entirely miss the surface and not return a height. In this case the height is set to zero, which overestimates the curvature, resulting in a jump of error as shown in Figure 6.6. At grid aligned angles the performance of GCHF is the best of all methods. Here the stencil is centred over the circle negating the previously discussed issue. GCNAHF, which is the best performing method with respect to overall error, while grid-centred, always has its stencil aligned with the surface normal meaning the stencil is well positioned to sample the surface.

Reviewing all the tests the GC methods had the best performance. GCHF performed

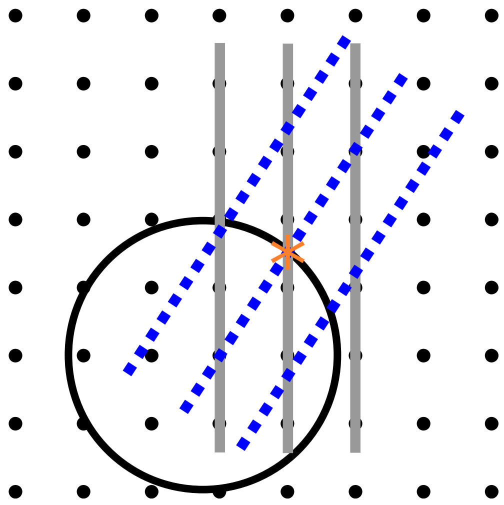


Figure 6.5: Curvature is requested at the orange $*$. HF uses a vertically aligned stencil, shown with solid gray lines, which results in the right most column missing the circle entirely. The NAHF stencil, shown with dashed blue lines, aligns better with the interface and all columns can record a valid height.

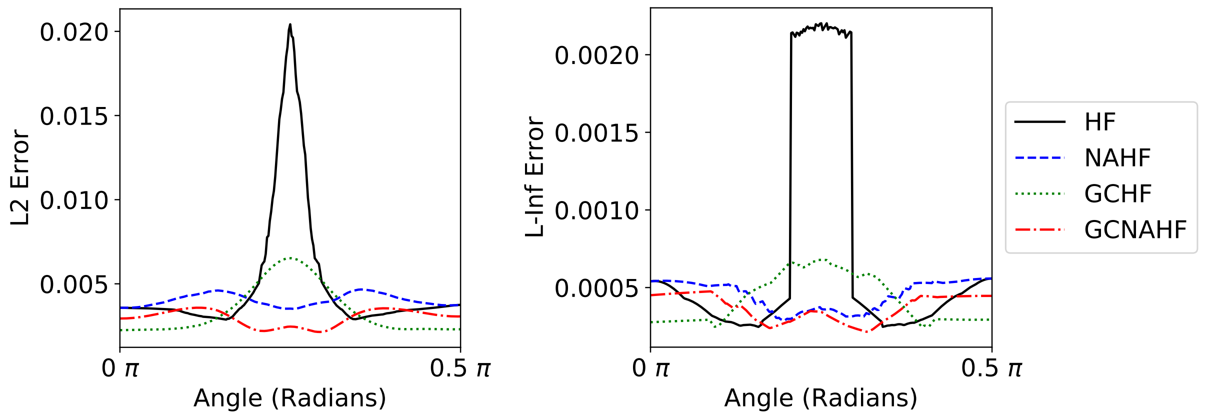


Figure 6.6: The errors of all height functions for high curvatures. All heights are measured using the iterative method except for GCHF. These simulations were performed with radii between 2 and 5 grid cells.

the best for low curvatures and GCNAHF for high curvatures. Normal alignment of the stencil does not seem to have a direct benefit using level sets. Instead it allows height functions to be successfully used on small geometries. Interpolation operations seem to be the most significant source of error given the improvement seen when using the GC methods. Note that in the work of Owkes et al. [42] they found that their normal-aligned method was only better for high curvature, low resolution data as well. They were using VOF surface tracking instead of level set which makes direct comparison difficult. They proposed a hybrid model to take advantage of their HF equivalent method that performed better on high resolution data. A similar idea could be applied here, switching between GCNAHF and GCHF. However, GCNAHF would be the best option if used exclusively due to its reliable performance on high curvature geometry, while still outperforming HF and NAHF on low curvature data.

6.2.2 Curvature Method Comparison for Direct Particle Levelset

A height function specific for DPLS was given in Section 4.5. Instead of using the level set to measure heights the actual particles tracking the surface are used. As previously shown, when height functions are used with level sets the interpolation error is the most significant problem. Using DPLS individual particles are selected eliminating interpolation error differences between methods. Therefore the GC family of methods are not relevant when using particles directly since there is no longer a grid present. This also revitalizes the possible benefits of NAHF beyond just better stencil alignment. However, the height functions from the previous section that use level set data are still valid for use on a DPLS. A level set is available, being reconstructed from the particles' positions. Therefore, we include in our comparison these methods as well.

Method	Low Curvature		High Curvature	
	L2	L-inf	L2	L-inf
HF	0.0157	0.144	0.0749	1.92
NAHF	0.0192	0.156	0.0570	0.293
GCHF	0.0115	0.145	0.0982	0.976
GCNAHF	0.0147	0.153	0.0615	0.284
HF-PLS	0.0118	0.163	0.0395	0.447
NAHF-PLS	0.00489	0.0281	0.0295	0.122

Table 6.7: Total L2 and L-inf errors for LS and DPLS height function methods. Low curvature is for circles with radii between 5 and 43 grid cells. High curvature is for circles with radii between 2 and 5 grid cells.

As before results are split into two groups: first high resolution, low curvature circles and lastly low resolution, high curvature circles. Throughout this section the DPLS specific methods are appended with "-PLS". Reviewing the results in Table 6.7 gives a conclusion on whether level set based height functions or DPLS based ones should be used. The HF-PLS method is competitive with the level set height functions while NAHF-PLS reduces error compared to the next best performing level set height function between 2-5$\times$ depending on the test and error metric. It should be noted that a full reseed cycle has taken place. The level set has been redistanced once, and the particles reseeded on that data, so that none of the methods can artificially benefit from an analytically initialized surface.

Figure 6.7 compares all height function methods for both pure level sets (Section 6.2.1) and those using DPLS. We observe an overall improvement for most methods when using DPLS. Note the only variable between these tests is the surface tracking method. HF and NAHF show an order of magnitude improvement for the low curvature tests. Clearly, NAHF-PLS integrates best with DPLS, but DPLS offers a more accurate representation of the surface than a standard level set as was shown in Section 6.1. This naturally benefits all methods with one exception. GCHF improves slightly on DPLS with low curvature data, and has slightly worse results on high curvature data. Since GCHF optimized interpolation error when using pure level set data it is not surprising that it benefits the least from a DPLS level set. Both GCHF and GCNAHF benefit from averaging multiple queries of curvature (which are theoretically identical for a circle) so their performance is artificially increased on pure level set data minimizing the effect that a DPLS level set has on their performance. GCNAHF does show a clear performance improvement on a DPLS level set.

The DPLS-specific methods can now be further examined. Their error as a function of angle is given in Figure 6.8. NAHF-PLS outperforms HF-PLS for both low and high curvature datasets. The better performance of NAHF compared to HF can now be observed since the interpolation error has been mostly negated through the use of particles. NAHF-PLS columns will pass through the interface at a more perpendicular angle than those of HF-PLS. A more perpendicular angle to the surface allows better selection of the seeded particle. Recall that the particles are seeded to the zero level set by being moved along the normal of the interface. Therefore seeding error displaces the particle along the interface normal. Displacement along the normal does not affect which particle is selected by NAHF since the distance between the column and the particle is perpendicular to the surface normal. Movement along the interface normal is not necessarily parallel to the column of a regular height function. Seeding error can result in the wrong particle being selected for HF-PLS, while for NAHF-PLS it just affects the measured height of the correct particle.

At low curvatures NAHF-PLS does not experience a jump in error reading curvature near the non-grid aligned $\pi/4$ angles. Note that at angles 0 and $\pi/2$ NAHF-PLS and

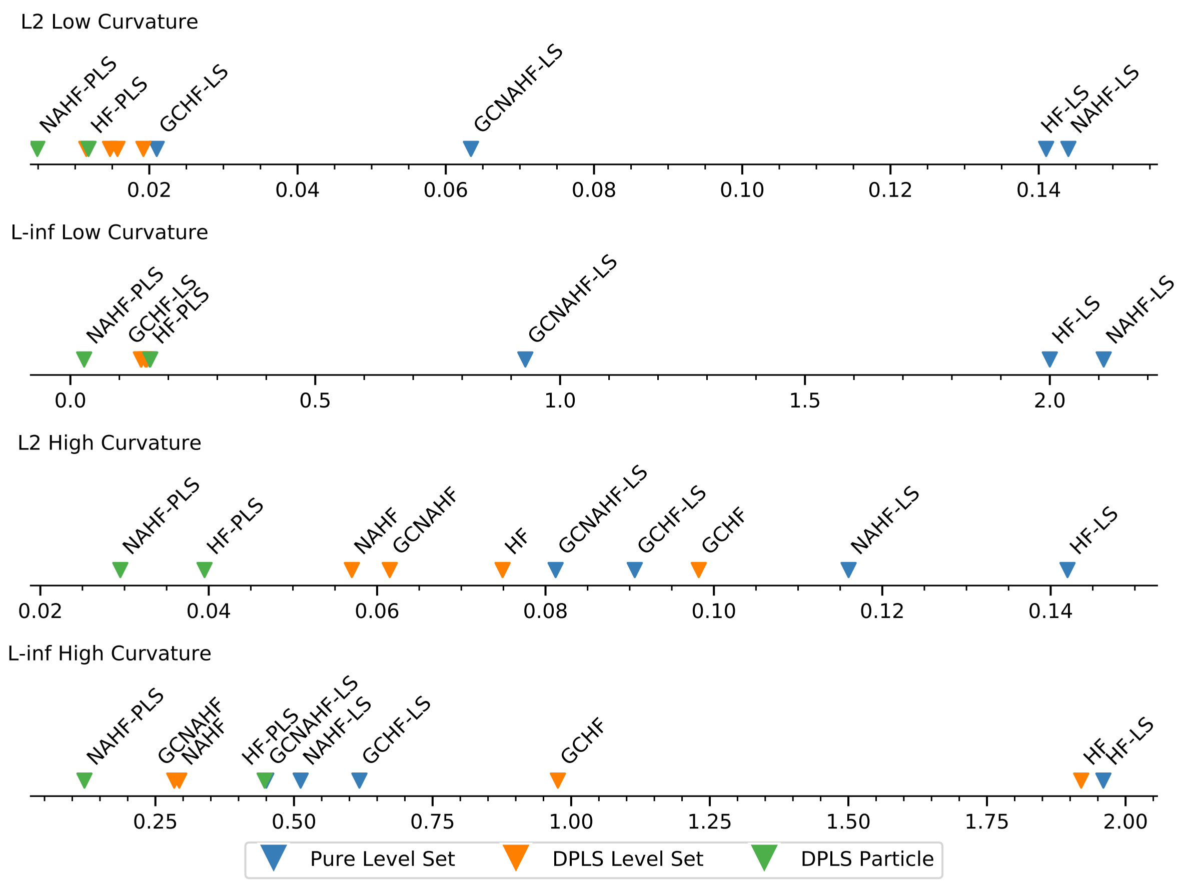
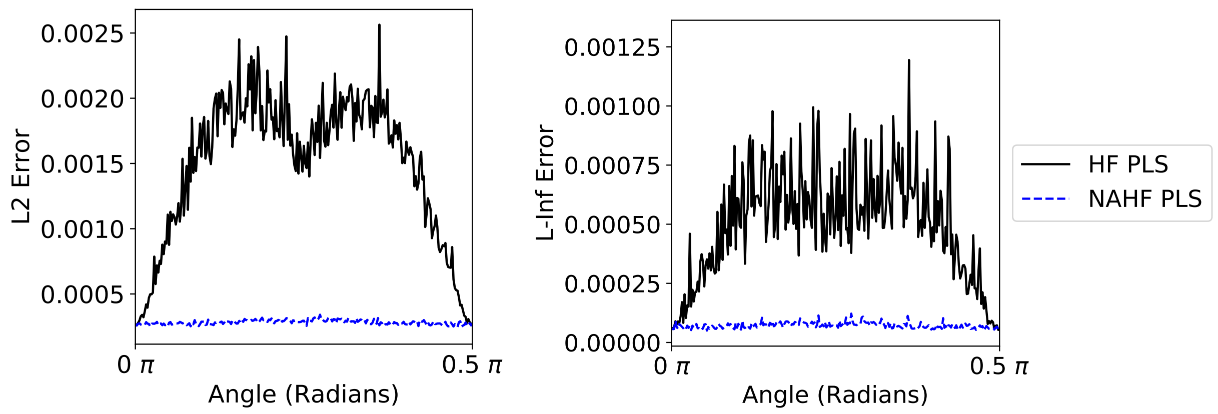
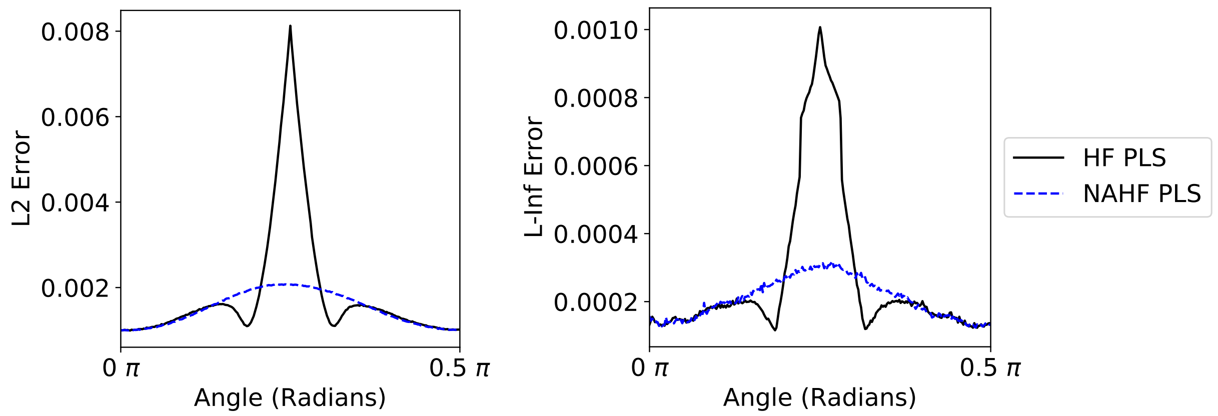


Figure 6.7: Curvature errors from Tables 6.5, 6.6 and 6.7 are plotted for comparison. "-LS" is appended to height functions operating on a pure level set from the previous section (6.2.1) and are denoted with blue markers. "-PLS" denotes height functions using DPLS particles directly to obtain heights as indicated with green markers. The remaining methods are using the level set generated from DPLS and are indicated with orange markers. For low curvature errors the labels of this last group are omitted due to similar performance.

HF-PLS have the same error. At these angles both methods are simultaneously normal and grid aligned. This shows that both implementations are operating as expected. For high curvatures NAHF-PLS benefits similarly to NAHF for pure level set data by having a better fitted stencil to small geometries. Near $\pi/4$ angles the HF-PLS error jumps due to poor stencil fitment on the small geometry. NAHF-PLS is therefore the best choice when



(a) Low curvature data (radii between 5 and 43 grid cells).



(b) High curvature data (radii between 2 and 5 grid cells).

Figure 6.8: HF-PLS and NAHF-PLS methods' errors as a function of angle.

using DPLS surface tracking. There is no need for hybrid methods such as found for level set in the previous section and by Owkes et al. for VOF. Lastly, NAHF-PLS on DPLS even outperforms all methods on level sets due to direct particle access and more accurate surface tracking.

6.2.3 Choosing Column Widths

It was mentioned in Section 4.16 that the stencil column spacing Δx_P does not necessarily have to be Δx. Owkes et al. [42] also experimented with different stencil widths and their results will be compared to the ones here. However their results were limited to measuring curvature on randomly generated circles. The constant curvature of circles, while convenient for testing, may not expose certain properties of height functions. Here another set of tests are introduced. The use of ellipses provide theoretically known non-constant curvature. Similar to the circles they are randomly generated except now the semi-major and semi-minor axis lengths have to be set. Both are randomly set, where the second one uses a uniform distribution centred on the value of the first.

The DPLS interface tracking method is used for testing since it reduces the effect of interpolation error. While particles are used, the grid resolution still affects how accurately the particles can be seeded to represent the circles initially defined on the underlying level set. This motivates the selection of a column width that best samples the heights without over or under sampling the data. The first set of tests are done with circles of varying locations and large radii to replicate some results from [42].

Circle Tests

Owkes et al. observed that large stencil widths (testing up to $\Delta x_P = 3\Delta x$) improve the accuracy of their NAHF for low curvature geometry. Therefore low curvature circles are used with radii between 47 and 50 grid cells to test if this implementation has the same property. This is approaching the upper limit for a 128 by 128 resolution grid while still allowing for some variance in position. In Section 6.2.1 it was shown that ill-fitting stencils perform poorly. Low curvature geometry may benefit from NAHF-PLS using a wider stencil provided it fits.

Δx_P	L2	L-inf
1.0	0.00259	0.0314
1.5	0.00172	0.0172
2.0	0.00134	0.00905
2.5	0.00121	0.00817

Table 6.8: Total L2 and L-inf errors as a function of stencil column width Δx_P. Dataset was composed of circles with radii between 47 and 50 grid cells.

The results of the tests are in Table 6.8. There is a decrease in both errors as stencil width is increased. As the stencil width is increased the height differences between columns also increases. However the height sampling error remains fixed, reducing the relative error.

For higher curvatures (radii between 15 and 40 grid cells) the result is no longer true as shown in Table 6.9.

Δx_P	L2	L-inf
1.0	0.00259	0.0275
1.5	0.00205	0.0146
2.0	0.00239	0.0109
2.5	0.00327	0.0119

Table 6.9: Total L2 and L-inf errors as a function of stencil column width Δx_P. Dataset was composed of circles with radii between 15 and 40 grid cells.

While all these stencil widths still fit the geometry there is a limit to beneficial stencil width. An optimum column width is now $1.5\Delta x$ or $2.0\Delta x$ if preference is given to L-inf error. A possible issue for stencils that are too wide is that they lose normal alignment of their left and right columns. Recall that only the centre stencil height column is truly ever normal-aligned. This also matches the results from Owkes et al. when they used large stencils on small geometries.

Ellipse Tests

Circles have constant curvature. Wider stencils may be artificially benefiting from this property as there is no penalty in accuracy due to missing surface curvature. In the reference frame of the normal-aligned stencil a circle's interface is always symmetrical. Using ellipses ensures that there is non-constant curvature. The ellipse axes are constrained between 15 and 40 grid cells for this test. As seen in Table 6.10 there is still a benefit to using a wider stencil than the minimum width.

Comparing Table 6.10 to the results in Table 6.9 a stencil that is too wide quickly is penalized due to the changing curvature of the ellipse. This may mean using a wide stencil is not applicable for real world data. However, given the benefit in accuracy for low constant curvature surfaces, the option to use a wider stencil would still be useful.

Inspired by these results we propose the use of a variable width stencil. Given an estimate of the curvature κ_E an appropriate stencil width can be chosen. Such an estimate can be achieved by using NAHF-PLS with $\Delta x_P = 1.0\Delta x$. First a relative curvature κ_R is

Δx_P	L2	L-inf
1.0	0.00271	0.0434
1.5	0.00239	0.0169
2.0	0.00312	0.0143
2.5	0.00451	0.0457

Table 6.10: Total L2 and L-inf errors as a function of stencil column width Δx_P. Dataset was composed of ellipses with the one axis set between 15 and 40 grid cells and the other randomly selected between 0.5 and 6.0 grid cells times the first.

calculated.

$$\kappa_R = \frac{\kappa_E}{\left(\dfrac{1}{\alpha \Delta x}\right)} \tag{6.8}$$

Fedkiw et al. [30] note that $1/\Delta x$ is the smallest curvature that can be represented by a grid. For the purposes of selecting stencil width the smallest stencil should be chosen well before the smallest curvature is reached. This is adjusted by the α parameter. In practise setting $\alpha = 8$ works well for all the tests presented here where $\Delta x_{Pmax} = 2.5\Delta x$ is the maximum stencil width. The stencil width is then calculated using Equation 6.9.

$$\Delta x_P = \left[1 + \min(\kappa_R, 1.0)\left(\frac{\Delta x_{Pmax}}{\Delta x} - 1\right)\right]\Delta x \tag{6.9}$$

Test	Best Fixed		Variable	
	L2	L-inf	L2	L-inf
Low Curvature Circles	0.00121	0.00817	0.00124	0.00783
High Curvature Circles	0.00205	0.0109	0.00223	0.0119
Ellipses	0.00239	0.0143	0.00263	0.0135

Table 6.11: The best L2 and L-inf errors are taken for low curvature circles (Table 6.8), high curvature circles (Table 6.9) and ellipse (Table 6.10) fixed stencil width tests. The corresponding variable width stencil errors are listed.

Using this method may not always give optimal results compared to handpicking the stencil width. Shown in Table 6.11 are the errors of the variable stencil compared to the best L2 and L-inf errors from the fixed widths. In many of these cases the best L2 and L-inf errors come from different stencil widths. The variable method is left unchanged between tests, as if being used with unknown geometry. The variable method is able to closely match both error metrics since it can dynamically change its width depending where it is sampling the ellipse. The errors of the variable width stencil are always smaller than the errors of the second best performing fixed stencil width, while not always matching the lowest observed individual L2 and L-inf errors. The main benefit is a variable width stencil eliminates the need to hand pick a stencil width.

6.2.4 Summary

We first demonstrated that the Laplacian type curvature measurement does not work well in practise on redistanced data. Height functions were shown to be less affected by redistancing, with an error one order of magnitude less than the Laplacian method. Testing a variety of height functions we determined that the normal-aligned types do not reduce error for a plain level set, except when due to better stencil alignment. However, normal alignment does show a significant reduction in error, for all tests, compared to axis alignment when integrated with DPLS surface tracking. Using DPLS with NAHF-PLS resulted in the lowest error for all tests, hence it will be used for the fluid simulation in subsequent sections. Our study of optimal column widths adds to the work of Owkes et al. [42] showing that the optimal stencil width depends on the geometry being measured. We proposed a simple column width estimate that closely matches the errors of stencils with handpicked widths. Since the ferrofluid simulation generates small features relative to grid resolution a column width of Δx is used instead of Equation 6.9.

6.3 Fluid Simulation

Our ferrofluid simulation will first be verified for non-magnetic fluids because a failure to simulate these precludes the successful simulation of a ferrofluid.

6.3.1 Verification of Surface Tension

Having verified a number of curvature measurement techniques, surface tension can now be tested. Surface tension applies a force such that the subsequent velocity moves the interface to minimize the curvature. In a zero gravity environment a successful surface tension model will evolve the interface towards a circle in 2D or a sphere in 3D.

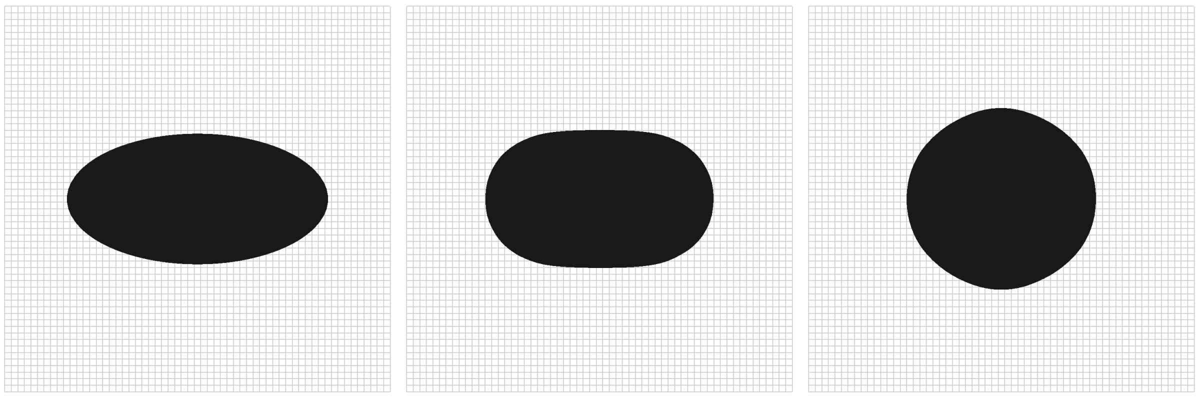
Therefore, the surface tension model here is verified by initializing the fluid as an ellipse. The fluid is placed in a zero gravity environment. Surface tension is the only force to cause initial motion of the fluid. The fluid should evolve towards a circular shape. The ultimate combination of DPLS surface tracking and NAHF-PLS curvature measurement is used. Viscosity is added to remove energy so that the simulation resolves to a circle, rather than infinitely oscillating. The simulation results are given in Figure 6.9 as select key frames.

The simulation begins at time step 0, with the fluid initialized as an ellipse. Immediately the semi-major axis begins to shrink as the surface tension starts to minimize the high curvature at the ends. Time step 250 shows an intermediate step of this occurring. By time step 485 the fluid is now a circle, but due to the momentum imparted previously the semi-major and semi-minor axis switch. The circle becomes an ellipse stretching along the vertical direction. This continues until time step 1190, at which the stretching stops and the ellipse begins to approach circular form again as shown in time step 1520. The process is a decaying oscillation completing the first period at time step 2100. The viscosity reduces the total fluid's energy decreasing the maximum curvature and semi-major axis length at the end of each period.

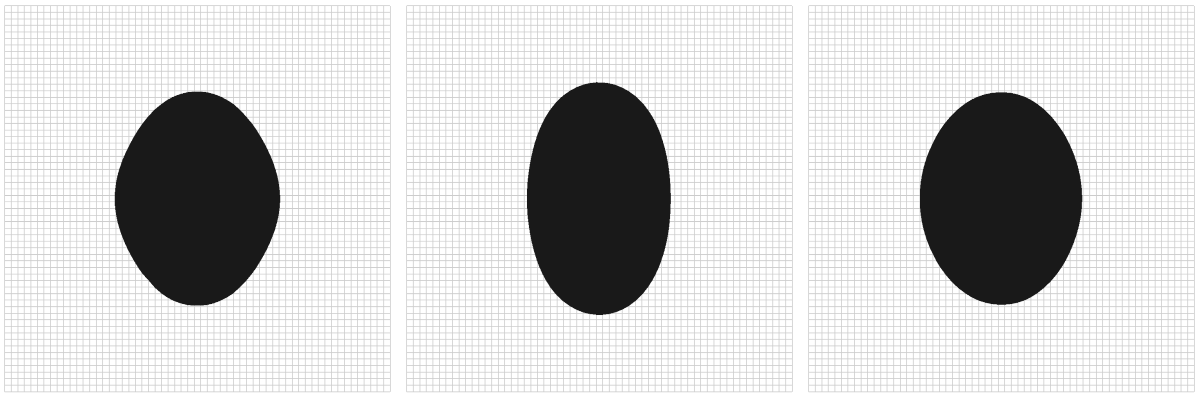
Shown in Figure 6.10 is the final simulation frame (after 11173 time steps). As expected the oscillation has decayed due to viscosity and the surface tension forces attain an equilibrium state by forcing the fluid to be a circle.

Another type of surface tension test involves initializing a cube in the centre of the domain (for a 3D simulator). There is no initial velocity or gravity applied. In the absence of surface tension no forces act to disturb the shape, and the cube would remain stationary. However with surface tension, the sharp curvature at the corners of the cube results in forces that cause a gradual transition to a sphere. This behaviour can be observed in Figure 6.11.

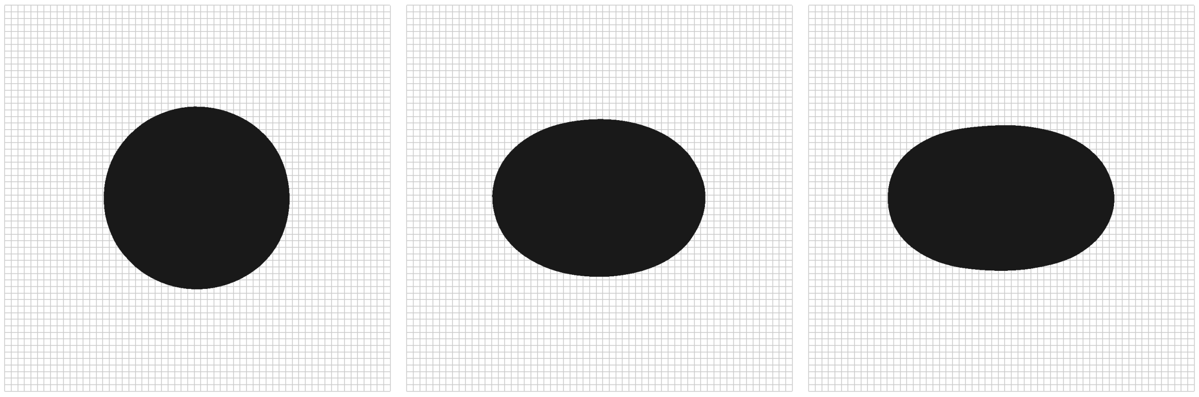
Immediately the sharp edges of the cube begin to round at the start of the simulation, starting the cube's transition to a sphere. This test is fairly trivial for surface tension using the Laplace style curvature discussed in Sections 6.2 and 4.5.1 since it is calculated directly from the level set. However, transitioning a cube to a sphere is an excellent stress test for height function based curvature measurement since it shows that the stencil can be correctly placed on areas of extreme curvature. Measuring curvature at the edges of a cube motivates the use of the NAHF method since the stencil sits at a $\pi/4$ angle to the side of the cube allowing for all height columns to intersect the cube's surface. With standard axis-aligned height functions (on an axis-aligned cube) some columns of the stencil would never intersect the cube surface requiring an additional case to prevent returning an infinite value for curvature. Strictly speaking, this is correct for a cube, but is not desirable computationally.



(a) Horizontal contraction during time steps 0, 250, and 485.



(b) Vertical expansion and contraction during time steps 650, 1190, and 1350.



(c) Horizontal expansion during time steps 1520, 1700 and 2100.

Figure 6.9: Oscillation of an ellipse in zero gravity due to surface tension.

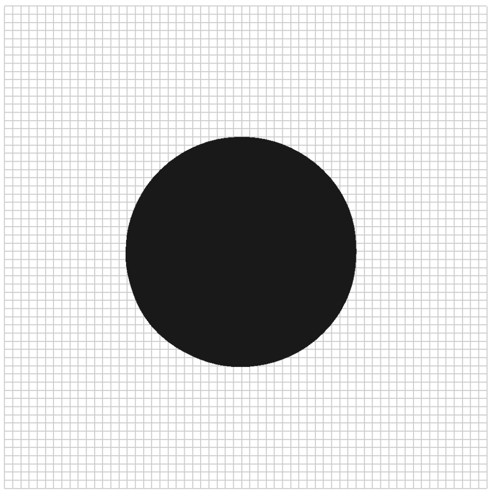


Figure 6.10: Final simulation result. Oscillation has stopped, leaving the fluid as a circle.

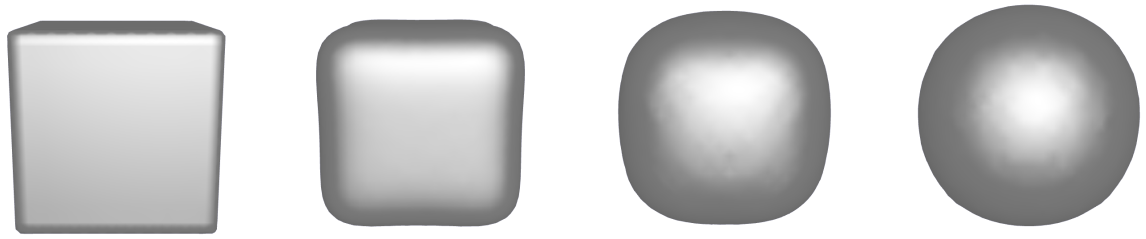
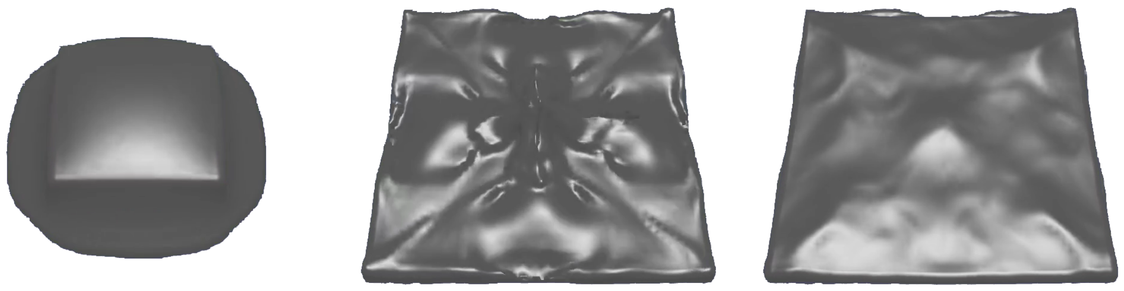


Figure 6.11: Cube evolving into a sphere due to surface tension. Viscosity is present to prevent oscillation.

These tests provide a qualitative verification of the surface tension algorithm. This demonstrates the successful integration of our curvature methods with the fluid simulator.

6.3.2 Free Fall

To verify that gravity and pressure can form an equilibrium a free fall test is performed. A fluid dropped from air into a containing vessel will accelerate due to gravity eventually hitting the vessel's bottom. A pressure forms to counteract the force of gravity preventing the fluid from moving through the bottom wall. Following basic fluid properties the fluid should flow outwards and fill the container. Viscosity should then cause the fluid to lose energy and eventually come to rest. The results are presented using the 3D simulator's output. Select frames are show in Figure 6.12.


(a) Free fall frames 15, 100 and 300 without viscosity.

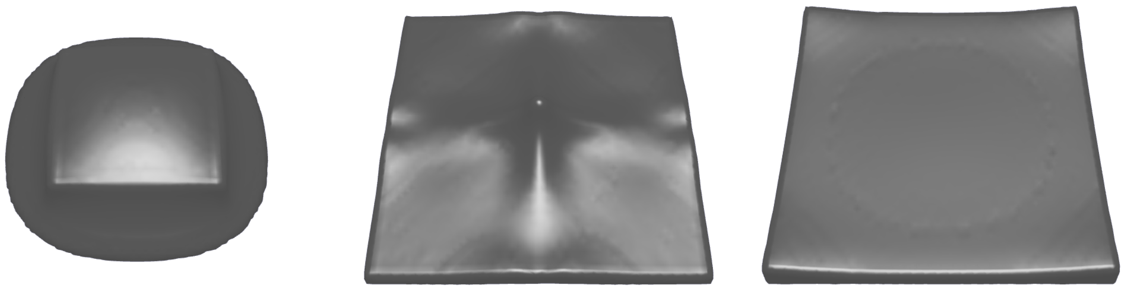

(b) Free fall frames 15, 100 and 300 with added viscosity.

Figure 6.12: Freefall of fluid in 3D. The fluid was initialized as a cube. The effect of viscosity is shown. Simulation resolution was 60^3.

The fluid is initialized as a cube centred above the bottom of the domain's solid lower bound. The simulation starts and the fluid accelerates downwards. As expected, when the fluid contacts the bottom of the domain a spreading behaviour is observed. This is shown in frame 15 of Figure 6.12. The fluid continues spreading until it contacts the edges of the domain. The fluid then travels back towards the centre of the domain forming a jet there as seen in frame 100. A wave occurs halfway between each wall in the viscous version as the fluid that reflects off the walls comes back and collides midway. The effect of viscosity is clearly seen, with the viscous version forming clear waves and just one jet in the centre. The inviscid version allows for small waves to form. This comparison continues for frame 300 where the viscous version has nearly come to an equilibrium, while substantial activity is observed for the inviscid fluid. Both sets of results show good symmetry throughout all frames. This is expected as the initial conditions and the container are symmetrical.

For this example running DPLS surface tracking on GPU yielded a twenty times speedup on average when advancing the level set. When a step called for a reseed of particles the performance improvement jumped to over thirty times that of CPU. The performance improvements should continue to increase with the number of particles used since GPU-specific overheads would become less significant. Note that the CPU DPLS implementation includes multithreaded particle advection.

6.4 Ferrofluid

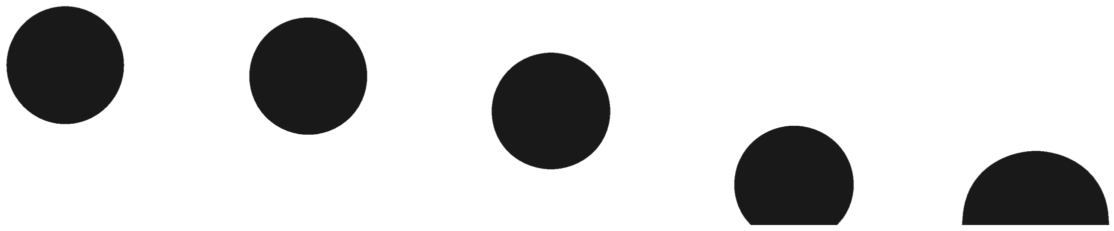
Finally the magnetic components of the simulation can be demonstrated. Having verified the fluid components of the simulation any additional behaviour observed is due to the unique magnetic nature of the ferrofluid. The simplest test is to see that the ferrofluid responds to an applied magnetic field.

6.4.1 Field-Induced Motion

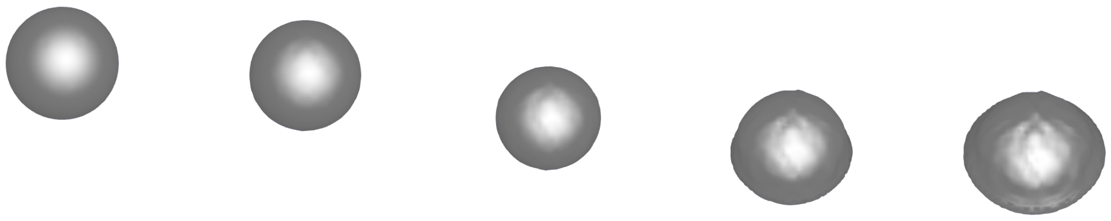
Since this simulator is based on the computational ferrofluid model of Afkhami et al. it should be able to reproduce their results [17]. A significant difference is their simulation was done with two fluid phases, that is, a ferrofluid moving in a non-magnetic fluid. Therefore some behaviour cannot be reproduced but the basic result from their field-induced motion of a ferrofluid droplet experiment can be recreated here.

The experiment is done in a zero gravity environment. A ferrofluid droplet is suspended in air with zero initial velocity. A magnetic dipole, as defined in Equation 5.3, is placed below the simulation domain. The ferrofluid droplet should experience a force that accelerates it towards the magnet. The results of this test are shown in Figure 6.13.

Experiments are done in both 2D and 3D simulators to verify the magnetic solver implementations. As observed the ferrofluid droplet moves towards the magnet below the



(a) Field-induced motion of a ferrofluid droplet in a zero gravity 2D environment. Time steps 1, 50, 100, 150 and 200 are shown. Resolution is 60^2 with a magnetic field between 40 and 199 A/m corresponding to the top and bottom of the domain respectively.



(b) Field-induced motion of a ferrofluid droplet in a zero gravity 3D environment. Time steps 1, 100, 200, 235 and 250 are shown. Resolution is 60^3 with a magnetic field between 36 and 386 A/m corresponding to the top and bottom of the domain respectively.

Figure 6.13: Field-induced motion of a ferrofluid droplet in both 2D and 3D simulators. Since the physics are not directly comparable as discussed in Section 5.1 the two results are not analagous due to differing applied magnetic fields. Time steps are constant at $\Delta t = 0.5s$ each and $\mu_r = 1.25$ for the ferrofluid.

domain. The magnetic force is kept lower than that used for creating the Rosensweig instability, hence the ferrofluid responds to the magnetic force in a similar fashion to gravity. However, the magnetic field is stronger at the bottom of the domain resulting in higher acceleration as the droplet moves down. Once the drop hits the bottom of the container it flows outward. This sequence of behaviour is also observed by Afkhami et al. [17]. However, due to our simulation being only single phase we are unable to reproduce droplet shape changes as there is no fluid surrounding the droplet. Nevertheless, this test shows that the magnetic potential solve, formation of the magnetic force tensor, and application of magnetic force produce a physically reasonable output when added to the fluid simulator. Additional testing shows that when $\mu_r < 1$, indicating a diamagnetic material repelled by magnetism, the droplet moves away from the magnet hitting the top of the domain. This does not simulate a ferrofluid's behaviour, since ferrofluid is attracted to magnets, but shows that the simulator can also handle diamagnetic materials if desired.

6.4.2 Rosensweig Instability

Mass motion of the ferrofluid was studied in the previous section. Here, the interactions of the ferrofluid with itself are presented. Namely, we will evaluate this simulator's ability to capture the Rosensweig instability. A typical experiment starts with the ferrofluid resting in the bottom of a container. A magnetic field is then applied, usually with the source below the ferrofluid. Different strengths and types of magnetic field change the behaviour of the ferrofluid.

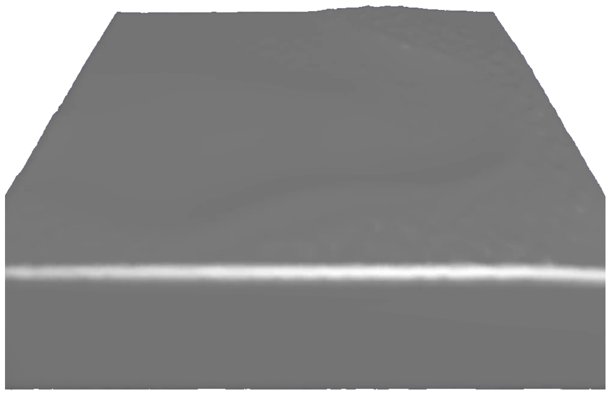
Critical Magnetization

The ferrofluid experiences a force at all magnetic field strengths. However, the Rosensweig instability can only form above a critical magnetization M_c or critical field H_c of the fluid as defined in Equations 3.21 and 3.22. In Section 6.4.1 the critical field was purposely kept less than H_c to avoid creating the instability. We now demonstrate that the simulator reproduces this critical field behaviour in Figure 6.14.

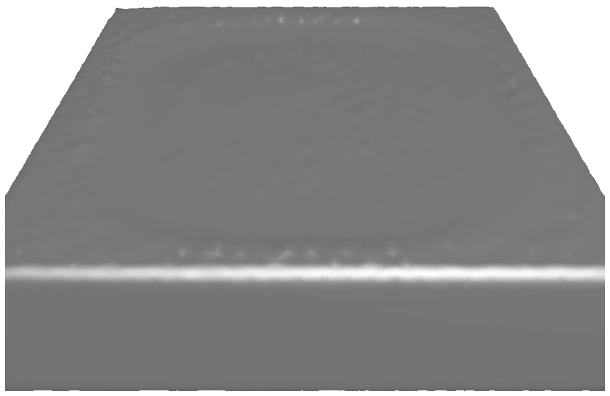
The magnetic field is gradually increased until the Rosensweig instability is observed. In each trial the fluid starts resting in the bottom half of the domain. A dipole magnet is centred 20cm below the fluid. The field decreases according to $1/r^3$ and is strongest along the central vertical axis of the domain. Recall that H_c is only a lower limit for the instability to form, it may not be sufficient for a particular experiment. For this experiment's parameters, as listed in Table 6.12, H_c is 7300 $\mathrm{A\,m^{-1}}$. These parameters are selected to maximize both the development and stability of peaks. The magnetic saturation M_s is higher than a typical ferrofluid to encourage peak growth. The viscosity ν is high to filter temporally localized velocities that may cause a peak to become unstable. The surface tension σ is slightly higher than that for water. The simulation resolution is $60{\times}60{\times}30$, where 30 is the vertical resolution. The scale $\Delta x = 1/240$ yields a 25cm by 25cm by 12.5cm domain.

In Figure 6.14a the applied magnetic field is weaker at the centre than H_c, so no peaks are expected. Zero movement of the surface is observed, indicating that gravity and surface

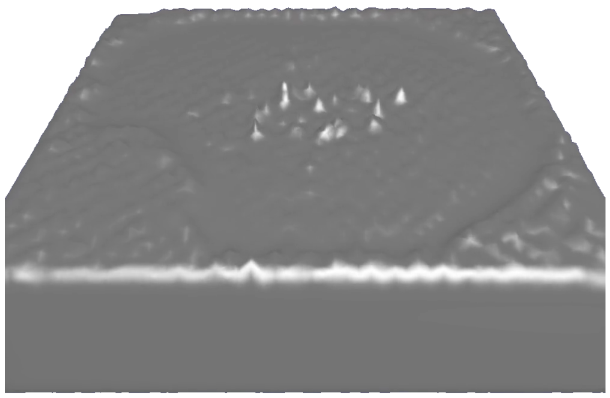
(a) m = 500 $\mathrm{A\,m^2}$, H = 2300 $\rightarrow$ 9900 $\mathrm{A\,m^{-1}}$



(b) m = 1500 $\mathrm{A\,m^2}$, H = 7000 $\rightarrow$ 30000 $\mathrm{A\,m^{-1}}$



(c) m = 1821 $\mathrm{A\,m^2}$, H = 8400 $\rightarrow$ 36000 $\mathrm{A\,m^{-1}}$



(d) m = 2250 $\mathrm{A\,m^2}$, H = 10000 $\rightarrow$ 45000$\mathrm{A\,m^{-1}}$

Figure 6.14: The critical magnetic field is found by gradually increasing the applied field until the Rosensweig instability starts to form. The parameter m denotes the magnetic moment of the magnet, and H is the magnetic field at the top and bottom of the domain respectively. Images are taken at 1.6s of simulated time.

tension are cancelling out any forces due to magnetism. This also demonstrates that the selected surface tension coefficient allows for a stable simulation.

Ripples on the fluid's surface can be observed in Figure 6.14b. The magnetic forces are now high enough to move the fluid, but not high enough to form the instability. Figure 6.14c starts to show the instability seeding itself in the centre of the domain. This is where the magnetic field is the strongest. There is also activity on the boundaries since their sharp curvature also serves as a location to seed peaks. However, this activity is below the

ρ $(\mathrm{kg\,m^{-3}})$	σ $(\mathrm{N\,m^{-1}})$	ν $(\mathrm{m^2\,s^{-1}})$	M_s $(\mathrm{A\,m^{-1}})$	m $(\mathrm{A\,m^2})$
1000	0.1	0.02	1824000	$2 \cdot 10^{-19}$

Table 6.12: Fluid parameters for the experiment in Figure 6.14.

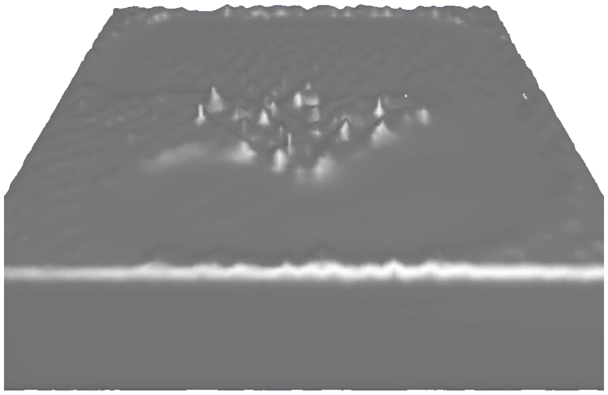
resolution of the simulator. The instability is first definitively observed in Figure 6.14d. The magnetic field is 19000 $\mathrm{A\,m^{-1}}$ as measured at the domain centre on the surface of the fluid. Peaking is localized to the centre of the domain since only here is the field stronger than the effective H_c. The effective H_c measured by our simulation is higher than the theoretical H_c of 7300 $\mathrm{A\,m^{-1}}$. The boundaries and high viscosity may be increasing the effective H_c since they are not accounted for in the derivation of Equation 3.22. The scale, resolution, time step, and surface tracking limitations of the simulator also increase the effective H_c value.

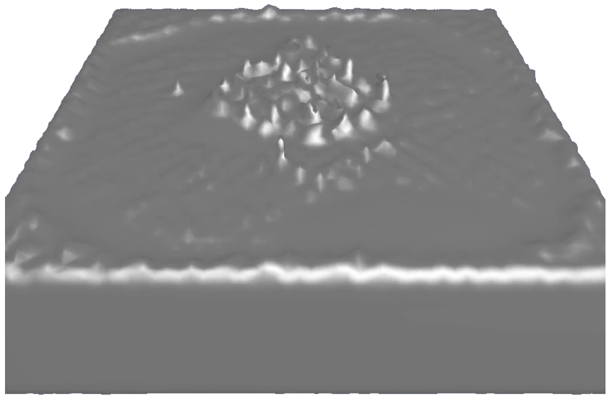
Observing the instability under the theoretical H_c would indicate that the simulator is artificially contributing to the instability. Whereas observing the instability over H_c may not indicate any problems due to the limitations of the theoretical basis for its derivation. At worst, it indicates that there are shortcomings to the simulator that resist the formation of the instability. This test of critical magnetization is therefore a good indicator that the simulator is producing the instability due to its physical model of ferrofluids and not from other sources.

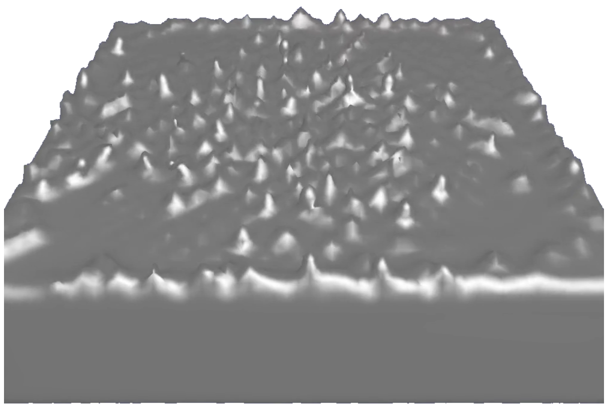
Magnetic Field

Once the instability has started, further increasing the applied magnetic field continues to grow the instability. Recall that the bifurcation parameter from Equation 3.23 is now $\epsilon > 0$ since $H > H_c$. The peak height should therefore grow as the magnetic field is increased as predicted by Equation 3.24. The critical field experiment from Figure 6.14 is continued in Figure 6.15 to show this property.

Starting with Figure 6.15a the magnetic field has further increased from Figure 6.14d. The area of peaking is increased since the region where $H > H_c$ on the fluid's surface is larger. Figure 6.15b continues this trend, with an expanded area of peaking. Figure 6.15c and 6.15d shows peaking over most of the fluid's surface. Note that the surface near the four corners of the domain displays minimal peaking since the magnetic field is the smallest there. Figure 6.15d has taller peaks than Figure 6.15c and much more noticeably so than in Figure 6.15a. There are also sizable peaks forming along the walls of the domain due


(a) m = 2360 $\mathrm{A\,m^2}$, H = 11000 $\rightarrow$ 47000$\mathrm{A\,m^{-1}}$


(b) m = 2570 $\mathrm{A\,m^2}$, H = 12000 $\rightarrow$ 51000$\mathrm{A\,m^{-1}}$


(c) m = 2680 $\mathrm{A\,m^2}$, H = 12400 $\rightarrow$ 53000$\mathrm{A\,m^{-1}}$

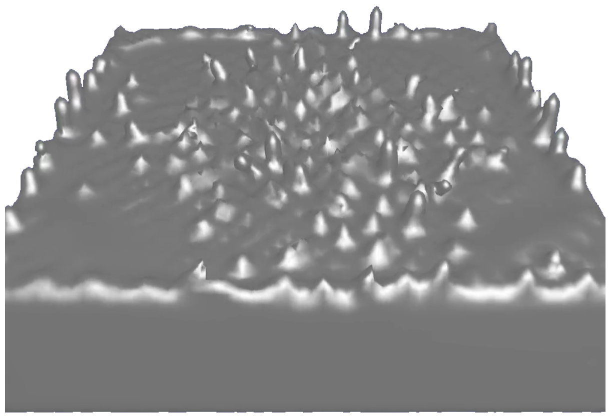

(d) m = 3000 $\mathrm{A\,m^2}$, H = 14000 $\rightarrow$ 60000$\mathrm{A\,m^{-1}}$

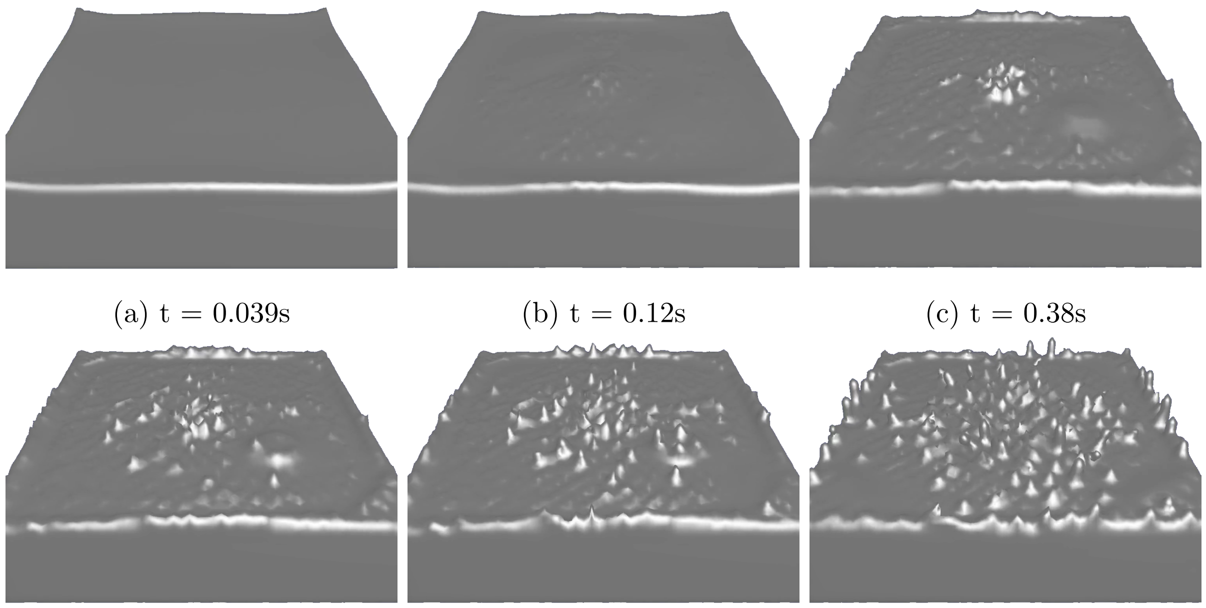
Figure 6.15: Continuation of experiment from Figure 6.14. The instability has already formed and is grown by increasing the applied magnetic field. The parameter m denotes the magnetic moment of the magnet, and H is the magnetic field at the top and bottom of the domain respectively. Images are taken at 1.6s of simulated time.

to the presence of the boundary along with an area of sharp curvature encouraging the growth of peaks. This is likely a numerical artifact.

This result confirms that peaking occurs primarily due to the presence of a sufficiently strong magnetic field. In addition, increasing the magnetic field creates more peaks, and eventually taller peaks as well. Both of these trends follow our expectation of how a ferrofluid should respond to a magnetic field. This supports the hypothesis that the simulator is using a correct physical model.

Time Dependence

The effect of increasing the magnetic field has been presented after the ferrofluid surface has already come to equilibrium. Each of these simulations also has time-dependence. As an example the simulation from Figure 6.15d is shown in Figure 6.16 as a function of time.


(a) t = 0.039s (b) t = 0.12s (c) t = 0.38s

(d) t = 0.54s (e) t = 0.78s (f) t = 1.7s

Figure 6.16: Select frames from the simulation that produced the final result in Figure 6.15d are shown as a function of time.

The simulation starts as a fluid at rest in the bottom half of the domain. The fluid responds to the magnetic force immediately after the start of the simulation as seen in Figure 6.16a. Note the raised corners of the fluid. No instability is present at this point. However, the start of it can be observed in Figure 6.16b, primarily at the centre of fluid surface since the magnetic field is the strongest there.

Figure 6.16c now shows definitive peaks at this location. Peaks are starting to form away from the centre of the domain as well, but are less developed as the total impulse is less. Observed in other ferrofluid simulations is a lack of symmetry [12]. In this example the initial peaking occurs in the centre, but subsequent peaking occurs at random locations that break symmetry. Starting in this frame, and continuing in Figure 6.16d is a localized set of peaks on the bottom right of the surface. They eventually disperse into the pattern of peaks by the final time step in Figure 6.16f. As the simulation reaches an equilibrium the rate of peak formation decreases noting the jump of one second between the last two shown frames of this simulation.

Simulation Challenges

We have demonstrated that our simulator is able to model several qualitative trends and properties of ferrofluids. We were able to capture the critical magnetization property of ferrofluids. Additionally, our simulated Rosensweig instability grows as the applied magnetic field is increased. The peaks also evolve as a function of time, developing over multiple time steps before reaching an equilibrium.

We believe that two key limitations prevent our simulator from producing the well-ordered hexagonal pattern characteristic of ferrofluids as shown in Figure 1.1. First, a stable equilibrium of forces is necessary to form such a pattern. Any erroneous forces cause peaks to shift positions resetting the pattern formation process. An easy way to reduce these forces is to use a highly viscous fluid such as in the examples shown. Parasitic currents, due to slight imbalances between surface tension and pressure, cannot cause as much movement in one time step with viscosity present. This is not a problem unique to a ferrofluid simulation, but has been a general problem for fluid simulation [38], with recent progress [48]. In addition to surface tension, a ferrofluid simulation also requires the balancing of gravitational and magnetic forces for peak formation to occur. This is likely an even more delicate equilibrium to obtain.

There are also semi-implicit surface tension schemes such as by Hysing et al. [37], or ones that relax the stability constraint (Section 4.5) such as by Sussman et al. [39]. Using these more stable methods could yield higher quality results. Popinet et al. [38] argues that these methods also damp lower frequency modes. While not ideal, the Rosensweig instability is a high frequency phenomena, meaning these methods may still work well. Another method to improve equilibrium quality is to use smaller time steps, however this also increases the simulation time. Extending the simulation to operate over multiple GPUs and CPUs would aid in reducing the resulting runtimes. All the results in this section took approximately 21 minutes each to generate using an AMD Ryzen 1700x CPU and NVIDIA GTX1070 GPU.

Another barrier to pattern formation is the underlying resolution and grid of the simulation. A regular grid does not naturally match a hexagonal pattern of peaks at near-grid resolution. This necessitates using a higher resolution, which incurs significant computational cost. Especially considering that only the surface of the ferrofluid requires this additional resolution, increasing the resolution throughout the entire domain would be inefficient. The use of octrees [49], which allow for the refinement of the grid in desired locations such as on the fluid's surface, would be a useful extension of the present simulator. However transferring an algorithm for regular grids to an octree adaptive grid is not a trivial procedure. For example, resulting discretizations must handle T-junctions between grid refinement level transitions [50]. Therefore, the switch to using an octree adaptive grid is left as future work.

Chapter 7

Conclusion and Future Work

We developed a general 3D Eulerian ferrofluid simulator capable of approximately producing the Rosensweig instability. The model for ferrofluid magnetism published by Afkhami et al. [17], which has been previously only used to simulate droplet level phenomena under well defined conditions, was discretized and solved for general conditions. For the simulator to better produce the Rosensweig instability we developed a direct particle level set method, denoted DPLS. To improve surface tension a leading curvature estimation algorithm was analyzed and adapted for improved performance with the DPLS method.

The use of height functions was thoroughly analyzed for both level sets and DPLS. When used on data that has been redistanced, such as in a simulation, height functions greatly reduce the error when compared to the traditional Laplacian curvature method. Iterative searching to determine column height values is also shown to have a minor benefit. The use of normal-aligned height functions does not provide a direct improvement of error compared to axis-aligned height functions for level sets. Their main advantage is to improve the fit of stencils on small geometry, reducing error for these specific cases.

DPLS and the normal-aligned height function method were combined so that particle location data is used directly, rather than level set data, to determine the interface position. When using this combined method normal alignment benefits directly through better estimates of column heights as well as improving stencil fit on small geometry. The combination of normal-aligned height functions and DPLS gave the lowest error in all tests motivating their use for the ferrofluid simulations. Lastly, an optimal column spacing method was presented and verified for normal-aligned height functions on DPLS data. The usefulness of our combined surface tracking and curvature methods can certainly extend beyond just ferrofluid simulation. Any simulation that requires maintaining a high level of detail on the fluid interface could benefit from this approach. The surface tracking and curvature methods could be packaged as an independent library for use. The CUDA GPU implementation of DPLS allows for substantial performance gains as well.

The results presented from our ferrofluid simulator were focused on reproducing the Rosensweig instability for a dish of ferrofluid. The simulator itself is not limited by any assumptions regarding this initial configuration. While the results do not rival those by Huang et al. [1], there are several areas of improvement for our simulator. The most significant would be switching to an octree based adaptive grid. This would be a significant change, requiring every simulation step to be updated. However proving that the fixed resolution version works is an important step in this process. For any methodology e.g., viscosity [50], the adaptive version follows the implementation of a fixed one. Reformulating the most performance intensive methods to use multiple CPUs/GPUs would also allow for higher resolution simulations in reasonable time frames. This could still be coupled with an adaptive grid method for additional performance if necessary.

Simulating the Rosensweig instability requires a stable, detail-preserving simulation. The Rosensweig instability provides an excellent stress test of a simulator due to this. Besides contributing the first implementation of a 3D Eulerian-based general ferrofluid simulator that can produce the Rosensweig instability, the demands of ferrofluid simulation motivated the development of new surface tracking and curvature estimate methods. Since 1987 there has been multiple attempts at simulating the Rosensweig instability. Simulations have resorted to non-physical models, restrictive assumptions, or have been hindered by their computational performance to the point of only producing one peak. A simulator without these limitations was recently developed using particle based simulation [1], and our work has developed one using grid based simulation. Both methods can reproduce qualitative trends. The next challenge for simulating the Rosensweig instability will be to achieve quantitative accuracy under non-restrictive conditions.

References

[1] Libo Huang, Torsten Hädrich, and Dominik L. Michels. On the accurate large-scale simulation of ferrofluids. *ACM Trans. Graph.*, 38(4):93:1–93:15, July 2019.

[2] Joseph L. Neuringer and Ronald E. Rosensweig. Ferrohydrodynamics. *The Physics of Fluids*, 7(12):1927–1937, 1964.

[3] R. E. Rosensweig. *An Introduction to Ferrohydrodynamics*. Dover Publications, Inc., Mineola, New York, 2014.

[4] C. Scherer and A. M. Figueiredo Neto. Ferrofluids: Properties and applications. *Brazilian Journal of Physics*, 35(3A), 2005.

[5] D. B. Hathaway. Use of ferrofluid in moving-coil loudspeakers. *DB-Sound Eng. Mag.*, 13(2), 1979.

[6] Oscar Agertz, Ben Moore, Joachim Stadel, Doug Potter, Francesco Miniati, Justin Read, Lucio Mayer, Artur Gawryszczak, Andrey Kravtsov, Åke Nordlund, Frazer Pearce, Vicent Quilis, Douglas Rudd, Volker Springel, James Stone, Elizabeth Tasker, Romain Teyssier, James Wadsley, and Rolf Walder. Fundamental differences between SPH and grid methods. *MNRAS*, 380(3):963–978, Sep 2007.

[7] R. Bridson. *Fluid Simulation for Computer Graphics*. CRC Press, Boca Raton, Florida, 2nd edition, 2016.

[8] A.G. Boudouvis, J.L. Puchalla, L.E. Scriven, and R.E. Rosensweig. Normal field instability and patterns in pools of ferrofluid. *Journal of Magnetism and Magnetic Materials*, 65(2):307 – 310, 1987.

[9] O. Lavrova, G. Matthies, T. Mitkova, V. Polevikov, and L. Tobiska. Numerical treatment of free surface problems in ferrohydrodynamics. *Journal of Physics: Condensed Matter*, 18(38):S2657–S2669, sep 2006.

[10] Olga Lavrova, Gunar Matthies, and Lutz Tobiska. Numerical study of soliton-like surface configurations on a magnetic fluid layer in the rosensweig instability. *Communications in Nonlinear Science and Numerical Simulation*, 13(7):1302 – 1310, 2008.

[11] Christian Gollwitzer, Gunar Matthies, Reinhard Richter, Ingo Rehberg, and Lutz Tobiska. The surface topography of a magnetic fluid: a quantitative comparison between experiment and numerical simulation. *Journal of Fluid Mechanics*, 571:455–474, 2007.

[12] Yuan Cao and Z.J. Ding. Formation of hexagonal pattern of ferrofluid in magnetic field. *Journal of Magnetism and Magnetic Materials*, 355:93 – 99, 2014.

[13] G. Yoshikawa, K. Hirata, F. Miyasaka, and Y. Okaue. Numerical analysis of transitional behavior of ferrofluid employing mps method and fem. In *Digests of the 2010 14th Biennial IEEE Conference on Electromagnetic Field Computation*, May 2010.

[14] Tomokazu Ishikawa, Yonghao Yue, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. Visual simulation of magnetic fluid taking into account dynamic deformation in spikes. In *IEVC2012*, 2012.

[15] Tomokazu Ishikawa, Yonghao Yue, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. Visual simulation of magnetic fluid using a procedural approach for spikes shape. In Gabriela Csurka, Martin Kraus, Robert S. Laramee, Paul Richard, and José Braz, editors, *Computer Vision, Imaging and Computer Graphics. Theory and Application*, pages 112–126, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[16] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[17] S. Afkhami, Y. Renardy, M. Renardy, J. S. Riffle, and T. St Pierre. Field-induced motion of ferrofluid droplets through immiscible viscous media. *Journal of Fluid Mechanics*, 610:363–380, 2008.

[18] S. Afkhami, A. J. Tyler, Y. Renardy, M. Renardy, T. St Pierre, R. C. Woodward, and J. S. Riffle. Deformation of a hydrophobic ferrofluid droplet suspended in a viscous medium under uniform magnetic fields. *Journal of Fluid Mechanics*, 663:358–384, 2010.

[19] Shahriar Afkhami, Linda J. Cummings, and Ian M. Griffiths. Interfacial deformation and jetting of a magnetic fluid. *Computers & Fluids*, 124:149 – 156, 2016. Special Issue for ICMMES-2014.

[20] Mingfeng Qiu, Shahriar Afkhami, Ching-Yao Chen, and James J. Feng. Interaction of a pair of ferrofluid drops inarotating magnetic field. *Journal of Fluid Mechanics*, 846:121–142, 2018.

[21] Nikolaos D. Katopodes. Chapter 13 - level set method. In Nikolaos D. Katopodes, editor, *Free-Surface Flow*, pages 804 – 828. Butterworth-Heinemann, 2019.

[22] D. J. Griffiths. *Introduction to Electrodynamics*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 3rd edition, 1999.

[23] R E Rosensweig. Magnetic fluids. *Annual Review of Fluid Mechanics*, 19(1):437–461, 1987.

[24] Bérengère Abou, José-Eduardo Wesfreid, and Stèphane Roux. The normal field instability in ferrofluids: hexagonsquare transition mechanism and wavenumber selection. *Journal of Fluid Mechanics*, 416:217–237, 2000.

[25] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[26] Mark Carlson, Peter J. Mucha, R. Brooks Van Horn, III, and Greg Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 167–174, New York, NY, USA, 2002. ACM.

[27] Henrik Fält and Douglas Roble. Fluids with extreme viscosity. In *ACM SIGGRAPH 2003 Sketches & Applications*, SIGGRAPH 03, page 1, New York, NY, USA, 2003. ACM.

[28] Christopher Batty and Robert Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA 08, pages 219–228, Goslar, Germany, 2008. Eurographics Association.

[29] Anthony Ralston. Runge-kutta methods with minimum error bounds. *Mathematics of Computation*, 16(80):431–437, 1962.

[30] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[31] Chris Wojtan, Matthias Müller-Fischer, and Tyson Brochu. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*, SIGGRAPH '11, pages 8:1–8:84, New York, NY, USA, 2011. ACM.

[32] Tyson Brochu and Robert Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput.*, 31(4):2472–2493, June 2009.

[33] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83–116, November 2002.

[34] Lanhao Zhao, Hongvan Khuc, Jia Mao, Xunnan Liu, and Eldad Avital. One-layer particle level set method. *Computers & Fluids*, 170:141 – 156, 2018.

[35] Mark Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.*, 187(1):110–136, May 2003.

[36] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA 04, pages 193–202, Goslar, Germany, 2004. Eurographics Association.

[37] S. Hysing. A new implicit surface tension implementation for interfacial flows. *International Journal for Numerical Methods in Fluids*, 51(6):659–672, 2006.

[38] Stéphane Popinet. Numerical models of surface tension. *Annual Review of Fluid Mechanics*, 50(1):49–75, 2018.

[39] M. Sussman and M. Ohta. A stable and efficient method for treating surface tension in incompressible two-phase flows. *SIAM Journal of Scientific Computing*, 31:2447–2471, 2009.

[40] S. Afkhami and M. Bussmann. Height functions for applying contact angles to 2d vof simulations. *International Journal for Numerical Methods in Fluids*, 57(4):453–472, 2008.

[41] S. Afkhami and M. Bussmann. Height functions for applying contact angles to 3d vof simulations. *International Journal for Numerical Methods in Fluids*, 61(8):827–847, 2009.

[42] Mark Owkes and Olivier Desjardins. A mesh-decoupled height function method for computing interface curvature. *Journal of Computational Physics*, 281:285 – 300, 2015.

[43] Stéphane Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838 – 5866, 2009.

[44] M. Yoon, G. Yoon, and C. Min. On solving the singular system arisen from poisson equation with neumann boundary condition. *Journal of Scientific Computing*, 69:391–405, 2016.

[45] Xu-Dong Liu, Ronald P. Fedkiw, and Myungjoo Kang. A boundary condition capturing method for poisson's equation on irregular domains. *Journal of Computational Physics*, 160(1):151 – 178, 2000.

[46] Alice Raeli, Michel Bergmann, and Angelo Iollo. A finite-difference method for the variable coefficient poisson equation on hierarchical cartesian meshes. *Journal of Computational Physics*, 355:59 – 77, 2018.

[47] Klaus Stierstadt and Mario Liu. Maxwell's stress tensor and the forces in magnetic liquids. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik*, 95(1):4–37, 2015.

[48] Moataz O. Abu-Al-Saud, Stéphane Popinet, and Hamdi A. Tchelepi. A conservative and well-balanced surface tension model. *Journal of Computational Physics*, 371:896 – 913, 2018.

[49] Frank Losasso, Ronald Fedkiw, and Stanley Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995 – 1010, 2006.

[50] Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. An adaptive variational finite difference framework for efficient symmetric octree viscosity. *ACM Trans. Graph.*, 38(4), July 2019.