UNIVERSITY COLLEGE CORK

AM4090 - FINAL YEAR PROJECT

SMOOTHED PARTICLE HYDRODYNAMICS: TOWARDS SALT DIAPIRS

Author:
DAVID MOLONEY

Supervisor:
DR KIERAN MULCHRONE

Contents

UNIVERSITY COLLEGE CORK

Abstract

Smoothed Particle Hydrodynamics: Towards Salt Diapirs

by David Moloney

Smoothed Particle Hydrodynamics (SPH) is a commonly used Lagrangian technique in the numerical simulation of fluid dynamics. Since its development in 1977 by Gingold and Monaghan [GM77], it has been extensively used in providing tests into scientific theories, offering insights into complex physical phenomena, and even in fluid models for video games. This project focuses on the development of an SPH method with the aim of modeling the growth of salt diapirs. It describes the applied mathematics behind the SPH technique as well as the various improvements made to code developed by Neill [Nei12]. The code is then used to provide a two-dimensional simulation of both a free flowing fluid, and the growth of salt diapirs.

UNIVERSITY COLLEGE CORK

Introduction

Smoothed Particle Hydrodynamics: Towards Salt Diapirs

by David Moloney

The initial title of the project was 'Numerical Modeling of the Development of Salt Diapirs', inspired by the work by Fernandez and Kaus [FK15]. The hope was to develop code to reproduce the results of the paper, and provide simulations of the development of salt diapirs in 3-D. In short, a diapir is a geological deformation found on the surface between two materials of contrasting densities. (A more detailed description of salt diapirs is discussed in section 3.2.1.)

A technique was established where a mix between an Eulerian, and a Lagrangian technique was to be implemented. A grid-based finite difference technique was to be used to solve for the velocity field of the system, whereas a particle-based Lagrangian method was to be used for the movement of salt 'particles' throughout the system with respect to the velocity field.

After much consideration, it was decided that self-developed code of such a complex nature was unlikely to be developed given the time frame. The code developed for use in Fernandez and Kaus [FK15] was obtained, but its complexity meant that it would be difficult to make significant progress towards fully understanding and reproducing their work.

In the end, it was decided that simplification of the problem to use a purely Lagrangian technique in 2-D would allow for significant progress, and Smoothed Particle Hydrodynamics was chosen soon after.

The aim of this project is to establish an SPH technique that can be used to produce similar results to those obtained in [FK15]. It was decided that C# code developed by Neill [Nei12] could be modified to simulate the development of Salt Diapirs. The first chapter of this report outlines the general formulation of an SPH technique. The second chapter describes the development of the program, including an overview of the algorithm used, as well as improvements made to the original code and difficulties encountered throughout the process. The final chapter outlines the final results of the project and its use in two examples; a single density free-flowing fluid, and a density contrast simulating the development of salt diapirs.

Chapter 1

The SPH Method

1.1 Fluid Mechanics

Often when hoping to model fluid mechanics, one must decide between an Eulerian and a Langrangian method. Essentially, an Eulerian technique tends to look at a fixed area, and tracks the fluid as it passes through, therefore looking at various fluid properties as a function of space and time. On the other hand, a Langrangian technique looks at fluid properties from the "fluid's point of view". Therefore, the fluid properties are viewed as a function of time alone.

Let's start by looking at the forces on a particular particle by considering Newton's second law:

$$\mathbf{F} = m\mathbf{a} \tag{1.1}$$

Note that in the classical study of fluid mechanics the Eulerian frame is usually taken. The acceleration **a** must be a derivative of the fluid velocity **u** that takes into account the movement of the fluid by both diffusion and advection. Therefore, the convective derivative is used.

$$\frac{\mathrm{D}\,\mathbf{u}}{\mathrm{D}\,t} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \tag{1.2}$$

By substituting into equation (1.1) above, and simplifying force terms, taking into account incompressibility will lead to the famous Navier-Stokes equation:

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \eta \nabla^2 \mathbf{u} + \rho \mathbf{g} \tag{1.3}$$

where $-\nabla p$ and $\eta \nabla^2 \mathbf{u}$ are pressure and viscous forces respectively.

Consider the Lagrangian approach to fluid mechanics by looking at the properties of individual fluid "particles". In this case, acceleration can simply be substituted for a time derivative of **u**. The result is the following momentum equation for particle i at position $\mathbf{r}_i$.

$$\rho(\mathbf{r}_i)\frac{\mathrm{d}\,\mathbf{u}(\mathbf{r}_i)}{\mathrm{d}\,t} = \mathbf{F}_i = -\nabla p(\mathbf{r}_i) + \eta \nabla^2 \mathbf{u}(\mathbf{r}_i) + \rho \mathbf{g}(\mathbf{r}_i) \tag{1.4}$$

Therefore, the force acting on a given particle can be broken up into forces due to pressure, viscosity, and external forces (usually gravity and boundary forces). This forms the basis for the SPH technique.

1.2 The SPH Approach

SPH or Smoothed Particle Hydrodynamics is a technique developed by Gingold and Monaghan [GM77] in 1977 for simulating fluid flow. The basic principle of this method is that the value of a physical quantity at position $\mathbf{r}$ is given by integral interpolation as follows:

$$A(\mathbf{r}) = \int A(\mathbf{r'})W(\mathbf{r} - \mathbf{r'}, h)\ \mathrm{d}\,\mathbf{r'} \tag{1.5}$$

where $W(\mathbf{r}, h)$ is a radial symmetric smoothing kernel with smoothing length h. In practice the smoothing kernel is an even function and normalised with $W(\mathbf{r}, h) = 0$ for $|\mathbf{r}| > h$. It acts as a quantity average over a circle of radius h with stronger weight given to nearby points. An example of a one-dimensional kernel is given in figure (1.1).

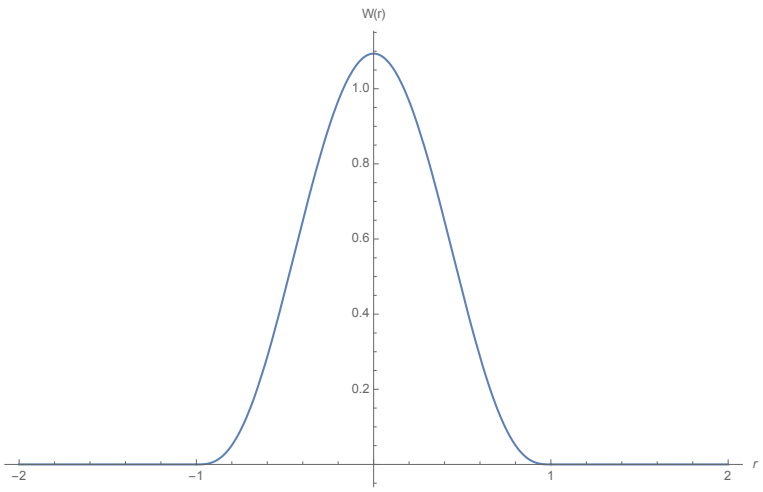

FIGURE 1.1: Order 6 polynomial smoothing kernel with $h = 1$

In the setting of fluid dynamics, the value of a quantity $A(\mathbf{r}_i)$ of particle *i* is given by

$$A(\mathbf{r}_i) = \sum_j A(\mathbf{r}_j)\frac{m_j}{\rho_j}W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.6}$$

which is considered a sum over nearby particles.

The gradient and Laplacian of particle properties can then be calculated as follows using a differentiable smoothing kernel. However, care must be taken when choosing the kernel so that the the gradient and Laplacian don't vanish at $\mathbf{r}_i = \mathbf{r}_j$ when required.

$$\nabla A(\mathbf{r}_i) = \sum_j A(\mathbf{r}_j) \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.7}$$

$$\nabla^2 A(\mathbf{r}_i) = \sum_j A(\mathbf{r}_j) \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.8}$$

Now the SPH equations can be used to formulate the forces in equation (1.4).

1.2.1 Pressure

Using the SPH equation (1.7), the standard form of the pressure force is defined as follows:

$$\mathbf{F}_i^{pressure} = -\nabla p(\mathbf{r}_i) = -\sum_j p_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.9}$$

Assuming the ideal gas state of the fluid as in [Aue09], scalar pressure of particle i is as follows:

$$p_i = k(\rho_i - \rho_0) \tag{1.10}$$

Where ρ_i is the particle-density (or particle concentration) at the position of particle i and ρ_0 is the overall rest density of the fluid. This scalar pressure can be thought of as follows. When there is a particle-density greater than the rest density, a positive pressure force is generated, pushing the particles apart. When there is a smaller particle-density than the rest density, a negative pressure force pulls the particles together. Therefore, equation (1.10) describes a scalar pressure that naturally attempts to bring the density of all particles to their rest density.

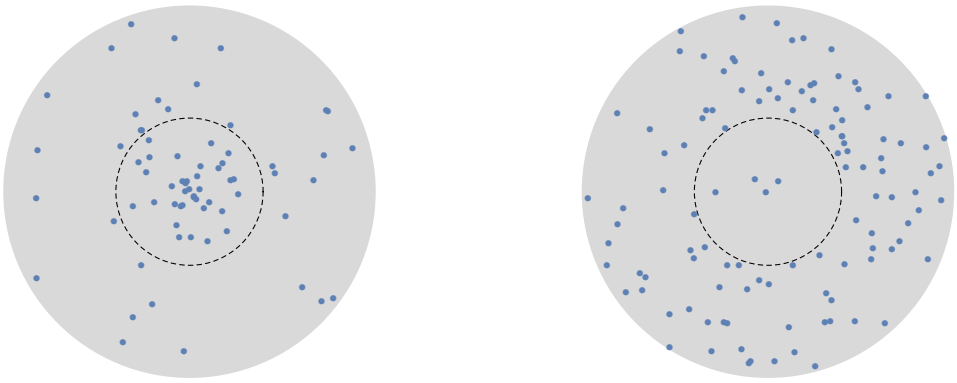

FIGURE 1.2: High and low particle-densities at the centre respectively.

The Tait equation proposed by Solenthaler and Pajarola [SP08] was also used to calculate the scalar pressure of particle i, with the hope of a more accurate model for simulations of systems with more than one type of particle.

$$p_i = \frac{k\rho_0}{\gamma}((\frac{\rho_i}{\rho_0})^\gamma - 1) \tag{1.11}$$

In equation (1.11), γ is an integer usually set to 7, and all other constants are defined as in equation (1.10). Note that by setting $\gamma = 1$, equation (1.11) can be seen as a generalisation of equation (1.10).

Both of the above equations require calculating the particle density at the position of particle *i*, which leads to the following section.

1.2.2 Density

The standard SPH equation for calculating the particle-density at the position of particle *i* is as follows:

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.12}$$

The above equation calculates the density by weighted average of the densities of the particles within the support of the smoothing kernel.

1.2.3 Viscosity

The viscosity force from equation (1.4) can be approximated using the standard SPH technique as follows:

$$\mathbf{F}_i^{viscosity} = \eta\nabla^2\mathbf{u}_i = \eta\sum_j \mathbf{u}_j \frac{m_j}{\rho_j}\nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{1.13}$$

1.2.4 External Forces

External forces are usually in two forms. In most cases forces due to gravity and boundary forces must be considered and applied to each particle. The force due to gravity is straightforward, but some thought must be given to the boundary forces. In both of the examples discussed in this paper, the boundary was constructed with immovable particles. Therefore, boundary forces need only be considered when a given particle is in close proximity to a boundary particle. This is a common technique discussed in detail in Liu and Liu [LL03]. Here the boundary force is given as:

$$\mathbf{F}_{ij}^{boundary} = \begin{cases} -D((\frac{r_0}{r_{ij}})^{n_1} - (\frac{r_0}{r_{ij}})^{n_2})\frac{\mathbf{x}_{ij}}{r_{ij}^2} & \text{if } r_o \leq r_{ij} \leq h \\ 0 & \text{otherwise} \end{cases} \tag{1.14}$$

In equation (1.14), n_1 and n_2 are usually taken to be 12 and 4 respectively. The value of D depends on the problem and is recommended to be of the same order of the square of the maximum velocity. r_0 is the cutoff distance, or the minimum distance from the boundary at which a boundary force will be applied, and in both of the examples is taken to be 4. r_{ij} is the distance from the boundary particle to the particle in question, and h is the usual smoothing radius. Finally, $\mathbf{x}_{ij}$ is the vector between the boundary particle, and the particle in question, with direction away from the boundary particle. Therefore, to calculate the boundary force to be applied to a given particle, the nearby boundary particles are summed over to accumulate the force.

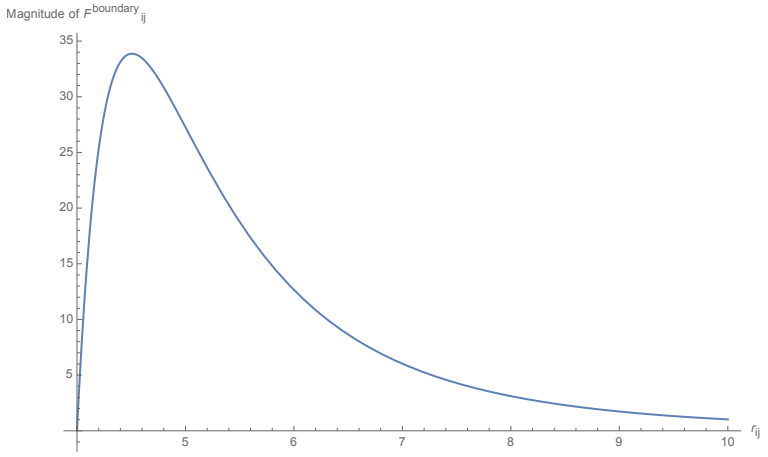

FIGURE 1.3: Magnitude of the boundary force on a given particle due to a boundary particle with distance r away.

Figure (1.3) shows the the magnitude of the force to be applied to a particle with distance r from a given boundary particle. For $0 \leq r_{ij} < r_0$ the magnitude of the force becomes negative and dominates all other forces to be applied to the particle, so to have sensible results the cutoff distance of r_0 must be applied. This provides the condition for the force in equation (1.14).

1.3 Integration Step

Now that the forces for a given particle have been calculated, a simple Forward-Euler approach can be implemented to move particles at each time step. At each time step, the velocity and position of each particle is updated as follows:

$$\mathbf{v}_{t+\delta t} = \mathbf{v}_t + \frac{\mathbf{F}}{m}\delta t \tag{1.15}$$

$$\mathbf{x}_{t+\delta t} = \mathbf{x}_t + \mathbf{v}_t \delta t \tag{1.16}$$

Chapter 2

Coding an SPH Program

2.1 Introduction

As mentioned previously, the code that was used in this project was developed from a similar SPH program developed by James Neill [Nei12]. This code was initially used to simulate elastic materials. The nature of the work in [Nei12] did not require a large number of particles. However, when hoping to simulate a free flowing liquid and a two-density system, a much larger system of particles was required. As a result, the efficiency of the program was much more important in this project. Consequently, the first task was to speed up calculations. This is discussed in section 2.3.

2.2 Pressure and Viscous Forces

Although the standard form of the pressure and viscous forces are defined in sections 1.2.1 and 1.2.3 respectively, the code that was used for this project used slightly different, but similar equations. Since there were few problems with these equations, it was decided not to change them to the standard form. Therefore, the pressure and viscous forces will be redefined in this section. The implemented pressure force as used by S. Clavet and Poulin [SP05] is defined as follows:

$$\mathbf{F}_i^{pressure} = -\sum_j (p_j W(\mathbf{r}_i - \mathbf{r}_j, h) + p_j^{near} W(\mathbf{r}_i - \mathbf{r}_j, h)^2)\hat{\mathbf{r}}_{ij} \tag{2.1}$$

In the above equation, p_j and the smoothing kernel are defined as before, and the choice of smoothing kernel will be specified in section 2.5. $\hat{\mathbf{r}}_{ij}$ is the unit vector going from particle j to particle i. p_j^{near} is called the near-pressure of particle j. It is introduced to add an extra pressure force in order to prevent particle clumping, an unnatural feature of many SPH programs. It is defined as follows:

$$p_i^{near} = k\rho_i^{near}$$

where

$$\rho_i^{near} = m_i \sum_j W(\mathbf{r}_i - \mathbf{r}_j, h)^2$$

Note that the smoothing kernel used in the definition of the near-density above is the same as the smoothing kernel used in the definition of the density in equation (1.12). It is not, however, the same as the smoothing kernel used in the newly defined pressure force equation (2.1).

As defined in [SP05], the implemented viscous force is defined as follows:

$$\mathbf{F}_i^{viscosity} = \sum_j (\sigma \mathbf{u}_{ij} + \beta \mathbf{u}_{ij}^2) W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.2}$$

In the above equation σ and β are viscosity coefficients, and $\mathbf{u}_{ij} = (\mathbf{u}_i - \mathbf{u}_j) \cdot \hat{\mathbf{r}}_{ij}$ is the velocity difference between particles i and j towards particle i.

2.3 The Neighbour-search Algorithm

At every time step the program must loop through all particles, and for each particle, create a list of neighbours (particles within a distance of the smoothing radius h). This is the most computationally expensive process of each simulation. The original program implemented an All-Pair search. That is, for each of the N particles, a loop over the remaining particles was implemented, and all particles within smoothing radius h were paired. So when checking for neighbours of the first particle, a loop over the remaining $(N-1)$ particles was carried out. Similarly, for the second particle in the list, a loop over the remaining $(N-2)$ particles was implemented. This method results in a search with complexity of order $O(\frac{N(N-1)}{2})$.

In order to speed up calculations a Linked-List search algorithm as described in [LL03] was devised. The first step in this technique is to divide the entire space into grids of width h, the smoothing radius. A section of such a grid is shown in figure (2.1).

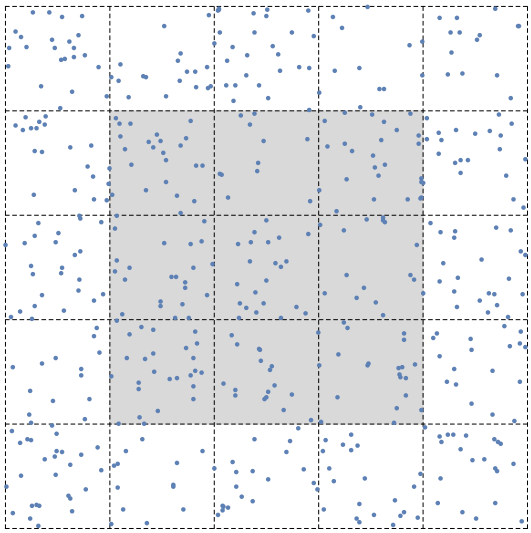

FIGURE 2.1: Shaded particle grid, through which the search algorithm is implemented.

It is important to note that for any particle in the centre grid in the diagram, any potential neighbour must lie in one of the nine shaded grids. Therefore, a new search algorithm can be developed by simply searching for particles in the nine surrounding grids. The general algorithm is as follows:

```
for Particle_i in Particle_List do
    for Near_Particle_j in Nearby_Particle_List do
        if r_ij ≤ h then
            Particle_i.NeighbourList.Add(Near_Particle_j);
        end
    end
end
```

Algorithm 1: Linked-List Search Algorithm

In *Algorithm 1*, for each particle, a loop through all particles in *Nearby_Particle_List* is implemented, where *Nearby_Particle_List* is a list of all particles in the surrounding grids. If a particle is found within the smoothing radius, it is added to the neighbour-list of particles of *Particle_i*. The complexity of this algorithm is dependent on the smoothing radius and the distribution of the particles throughout the grids. However, after completing some simulations with both search algorithms, it was found that the Linked-List algorithm was considerably faster, and it reduced running time by up to a factor of five.

2.4 Modifications to the Original SPH Code

As mentioned above, the original code was used to simulate elastic materials. Therefore, the original code made extensive use of spring-like forces between particles. In order to use this code to simulate a free flowing fluid all spring and elastic forces were removed. Therefore, the only forces to be considered were viscous, pressure, and external forces, as mentioned in the first chapter.

2.5 Choice of Kernel

A smoothing kernel, with smoothing radius h, is a real-valued function $W(\mathbf{r}, h)$ such that $W(\mathbf{r}, h) > 0$ for $|\mathbf{r}| < h$, and $W(\mathbf{r}, h) = 0$ for $|\mathbf{r}| \geq h$. Usually, it is required that:

$$\int_{-\infty}^{\infty} W(\mathbf{r}, h) \ \mathrm{d}\,\mathbf{r} = 1$$

The smoothing kernel must also be radially symmetric. That is, for any $\mathbf{r}_1$ and $\mathbf{r}_2$, such that $|\mathbf{r}_1| = |\mathbf{r}_2|$:

$$W(\mathbf{r}_1, h) = W(\mathbf{r}_2, h)$$

In the context of SPH, the smoothing kernel acts as a weighting function. For example, in the particle density equation (1.12), the smoothing kernel acts to give preference to nearest particles, and the more particles local to a point, the higher the particle density. Naturally, the choice of the smoothing kernel and the smoothing radius have a large impact on the results of an SPH simulation.

For the particle density, the smoothing kernel in the code was changed from an order-two spiked polynomial, to an order-six polynomial defined below, as used in [Aue09].

$$W_{poly6}(\mathbf{r}, h) = \begin{cases} \frac{A}{h^9}(h^2 - |\mathbf{r}|^2)^3 & \text{if } |\mathbf{r}| \leq h \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

In the above equation, a constant A is chosen such that the smoothing kernel integrates to 1.

For the pressure equation (2.1) and the viscosity equation (2.2) the following linear kernel was used. This choice of smoothing kernel was unchanged from the original code.

$$W_{linear}(\mathbf{r}, h) = \begin{cases} B(1 - \frac{|\mathbf{r}|}{h}) & \text{if } |\mathbf{r}| \leq h \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Similarly B is a normalisation constant.

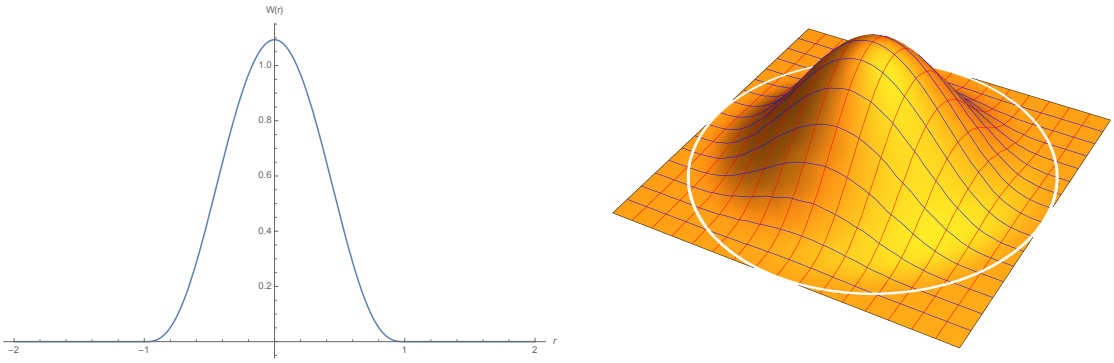

FIGURE 2.2: Order 6 polynomial smoothing kernel with $h = 1$. The first plot is simply a function of $|\mathbf{r}|$, the distance between particles. The second is the two-dimensional smoothing kernel.

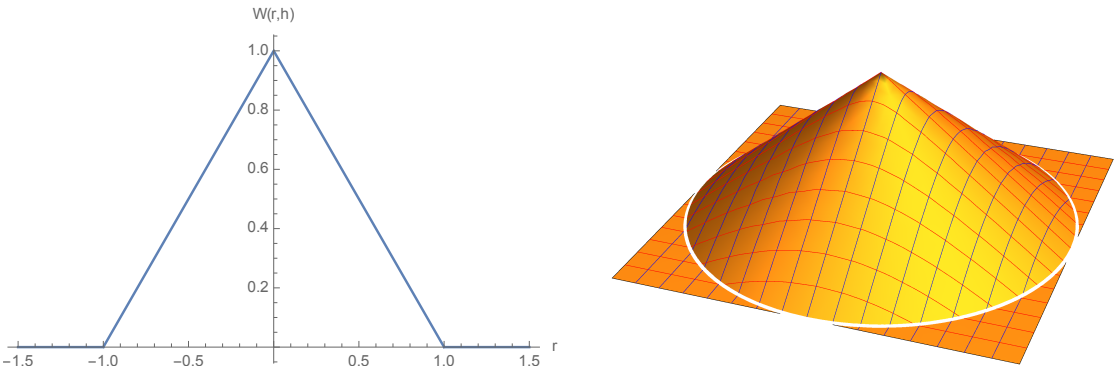

FIGURE 2.3: Linear smoothing kernel with $h = 1$.

2.6 Distribution of Particles

When first simulating the problem of the free flowing fluid, the initial particle distribution was set to be in a grid-like form. In other words, the particles were initially set in n rows of m particles with constant distance between them. This was in-line with the method used in the initial version of the code. A slightly unrealistic simulation resulted however with a relatively symmetric flow of particles.

Inspired by the technique used in [FK15], a more realistic approach was implemented by using a random distribution of particles in the initial setup. A simple uniform distribution for the x and y coordinate of each particle's initial position was used. This resulted in

a much more natural looking flow of particles, and also allowed for instabilities to occur between two types of particles with contrasting densities. This will be discussed further in sections 3.1 and 3.2.

However, use of a random distribution of particles does require some care when choosing both the number of particles and the bounds for the initial distribution of the particles. If too many particles are distributed into too small a space, this will result in an initial particle density much greater than the rest density of the fluid. In accordance with section 1.2.1, this will result in an initial pressure between the particles causing them to be pushed apart at the beginning of the simulation.

Similarly, if there are too few particles in the initial distribution space, the particle density will be lower than the rest density. This will result in the particles being pulled together, and in general the simulation will result in the fluid falling due to gravity until it reaches its rest density.

In order to reach a balance between the two situations described above, some trial and error experimentation was used until realistic results were achieved.

2.7 The SPH Algorithm

A concise, high-level algorithm for the overall SPH technique is described in the following pseudocode. At each time step, the neighbour-list of each particle is first calculated as described in Algorithm 1, and the general algorithm follows:

```
for Particle_i in Particle_List do
    Calculate_ParticleDensity(Particle_i);
    Particle_i.Force = gravity;
    for Near_Particle_j in Particle_i.NeighbourList do
        Particle_i.Force += Pressure_Force;
        Particle_i.Force += Viscous_Force;
        if Near_Particle_j.Boundary_Particle = True then
            Particle_i.Force += Boundary_Force;
        end
    end
    Particle_i.Velocity += Particle_i.Force / Particle_i.Mass * delta_t ;
    Particle_i.Position += Particle_i.Velocity * delta_t ;
end
```

Algorithm 2: General SPH Algorithm.

In the above algorithm, for each particle in the particle list, the particle density is calculated using equation (1.12). Following this, the force is set to gravity. For each particle in the neighbour-list, the pressure and viscous forces are accumulated, noting that the particle density is used in the calculation of the pressure force. Next, if a given neighbour particle is a boundary particle, a boundary force is added. Finally, the velocity and position of the particle is updated.

2.8 Issues

As to be expected when adapting code from a previous project, there were sections of code that caused some issues.

As mentioned in section 2.5, changes were made to the smoothing kernel used in the calculation of the particle density. As described in [Nei12], the stiffness and rest density in equation (1.10) were initially scaled to account for the linear smoothing kernel. This caused some difficulties as the scaling factor used is chosen so that the linear kernel integrates to 1. However, when the kernel was changed to a polynomial of order 6, the scaling factor needed to be modified. Therefore, the scaling in equation (1.10) was removed, and instead is found in the declaration of the kernel. This allows for changes in the kernel more easily.

The implementation of viscous forces caused much confusion for a large portion of the project. It was indicated in [Nei12] that the impulse due to viscosity was calculated, and the particle velocity was modified accordingly. However, the code seemed to add the viscous force to the accumulated velocity. This problem was resolved by finding a hidden scaling factor in time which converted the force back to impulse.

The form of the integration equations described in section 1.3 has caused some confusion throughout the project. In the original code, there seems to be a factor of δt missing when adding the accumulated force to both the velocity and the position in equations (1.15) and (1.16) respectively. However, it is expected that there is a hidden scaling of δt in the implementation of the force that would explain this issue. Given time constraints, the reasoning behind this issue was not found.

Chapter 3

Results

3.1 Free-Flowing Fluid

In order to test the modifications to the code, the SPH method was first applied to a single density fluid. A two dimensional box of immovable boundary particles was created, and a uniform random distribution of particles was placed in the box. The fluid was set to flow and settle naturally in various situations described below.

FIGURE 3.1: Free-flowing water.

The first, and most basic example is of a simple free-flowing system of particles. Two thousand particles are set with a random distribution in a rectangular box. The boundary particles are in red in the below images. The mass of each particle is set to 1.

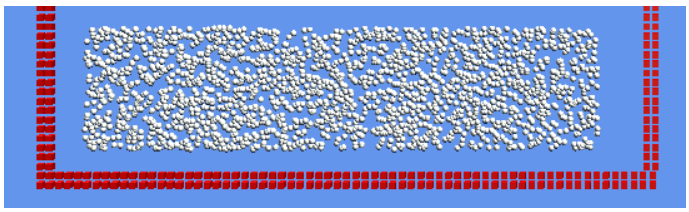

FIGURE 3.2: Initial distribution of the particles.

This example can be seen as water in a glass. Due to the random distribution of particles, the initial settling is due to the random gaps that appear between particles as

described in section 2.6. By changing the range of the particle distribution, the initial settling of the particles changes drastically. For example, having the same number of particles with twice the range in the y-axis will result in the particles "falling" initially over a greater distance.

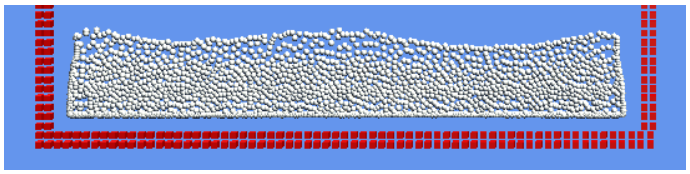

FIGURE 3.3: The particles begin to settle.

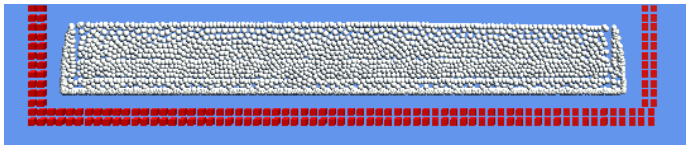

FIGURE 3.4: Settled particles, similar to still water.

Using the same setup, but with the Tait equation (1.11) for pressure, with $\gamma = 3$, and 1300 particles, the particles settle in a similar fashion.

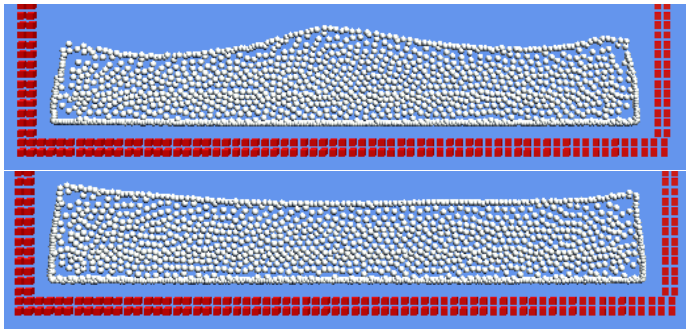

FIGURE 3.5: Fluid motion using the Tait equation for pressure.

It was found that the usual choice of $\gamma = 7$ resulted in pressure forces that were much too large. The Tait equation allows for fewer particles to be used as there is generally a larger pressure force keeping the particles further apart. However, the resulting motion seems very unstable. The fluid seemed to move unnaturally and would move around the box in response to a very small stimulus.

Extra care must also be put into the boundary forces when using the Tait equation. It was found that in many situations the larger pressure force causes particles to be pushed

through the boundary. It was decided that the original pressure equation gave more realistic results, so equation (1.10) was primarily used for the scalar pressure.

The next setup simply has the same two thousand particles, but with an initial distribution on the left side of the box. The particles are then released as before and allowed to settle. These could be seen as the flow of a fluid after a dam breaks.

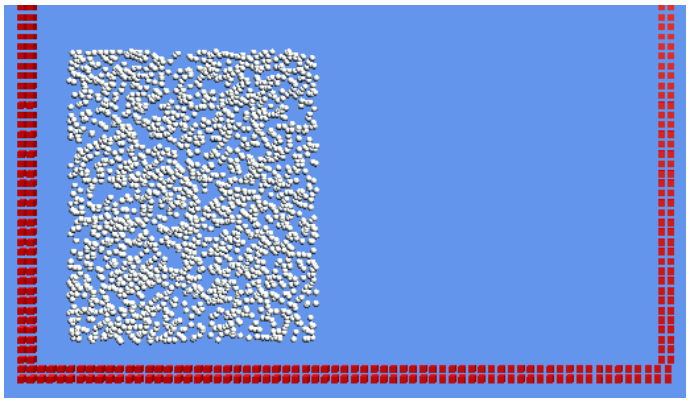

FIGURE 3.6: Initial particle distribution on the left side of the box.

The particles settle as expected, and in fact, model the movement of a liquid quite realistically as seen by the similarity with figure (3.1). However, it must be noted that in the third picture of figure (3.7), there is a slight unnatural gap once the wave breaks. This disappears almost immediately. It should also be noted that as a result of the large forces due to the crashing of the wave at the end of the first picture in figure (3.7), some particles will escape through the boundary with the setup as defined in previous sections. This could be prevented by increasing the multiplicative factor D in the boundary force equation (1.14).

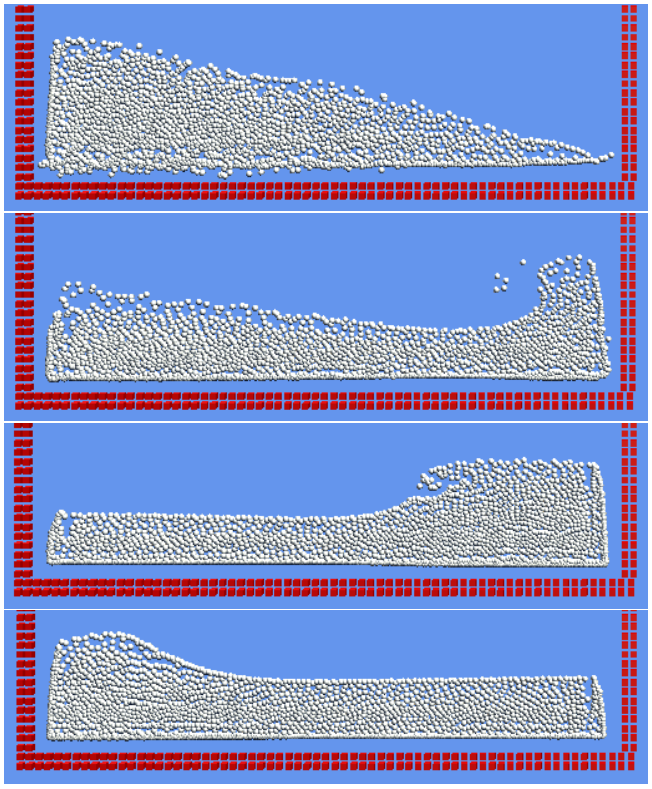

FIGURE 3.7: Particles flow throughout the box after release from the left.

The next example has the particles initially distributed on the left as before. However, the base of the box is sloped. This could represent a sloped bed on the edge of a lake.

As shown in figure (3.8), the sloped bed allows the particles to settle to have flat surface much quicker. The motion of the particles also causes the fluid to wash up and down the sloped bed, similar to that of waves washing up and down a beach.

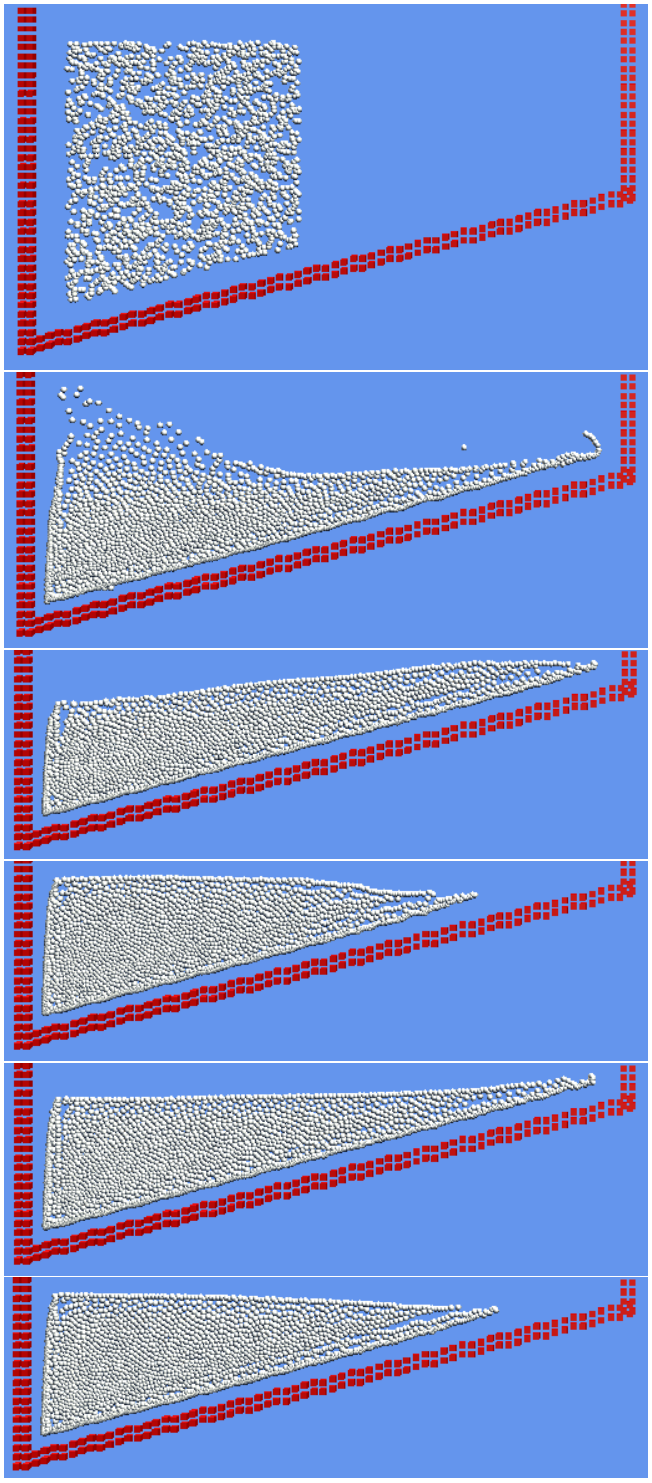

FIGURE 3.8: Particles flow up the sloped bed after release from the left.

3.2 Density Contrast

3.2.1 Description

The SPH technique can be generalised to model the development of salt diapirs due to natural materials of contrasting densities. In certain situations, the evaporation of salt-rich seawater from a marine basin over time can result in the deposition of salt in the ground. Over millions of years, natural sedimentation occurs, and the layer of sediment can build up and compress. This compressed sediment layer can grow in density to be in fact more dense than the salt layer. It is this density difference that causes the salt to be pulled up through the sediment, forming salt diapirs (or salt domes).

FIGURE 3.9: A model of Louisiana Coast Salt Dome.

In general, the salt diapirs appear in finger-like patterns as shown in figure (3.9). Usually the overlying sediment (rock) bends slightly as the salt moves through, giving it its dome-like shape. The bending in the sediment can create pockets between the rock and the salt, often containing natural gas and oil. Naturally, the development of salt diapirs attracts interest for commercial reasons.

3.2.2 Assumptions

The main assumption for the implementation of a two-density system to model salt diapirs is the ability to use fluids to model the motion of the salt. Both the salt and the

overlying sediment are in reality rock, so are inherently not fluid.

The primary benefit of the system is that the development of salt diapirs is a process that occurs over millions of years. Therefore, by using a very large time-step, the motion can be assumed to move like a fluid. This is an assumption commonly used in modeling geostatic movement within the earths surface, and in particular, is used in [FK15].

3.2.3 Modifications

The first and most obvious change made to the system was to implement a second density particle. This was accomplished relatively easily. The mass of one type of particle (Black) was declared with mass 1.3 times that of the other (White). The decision to choose 1.3 as the scale of the mass difference was not crucial. A larger ratio between the two densities simply results in a quicker formation of the diapir. In reality, the density of the sediment layer depends on the compressed nature of the rock. This varies depending on the situation.

Due to the random particle distribution, the particles will initially fall or push apart due to the initial particle-density of the random distribution. This results in a very unnatural wave-like motion for the first few seconds of the simulation. In order to force the system to settle, a boundary lid was put in place to restrict the wave-like motion.

3.3 Results

The initial setup was similar to that of the first example in section 3.1. 2300 particles were placed in a box with a lid to minimize the initial wave on the surface of the particles. The original pressure equation (1.10) was used.

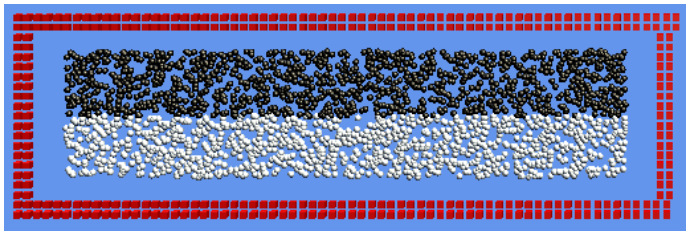

FIGURE 3.10: Initial distribution of the particles of different densities.

As mentioned previously, the initial few seconds of the simulation feature a wave-like behaviour due to the settling of the particles. This may cause an unnatural mixing of the particles.

As seen in figure (3.11), three main diapirs develop in this simulation, one of which in the centre is much narrower than the other two. If the simulation is run for longer the

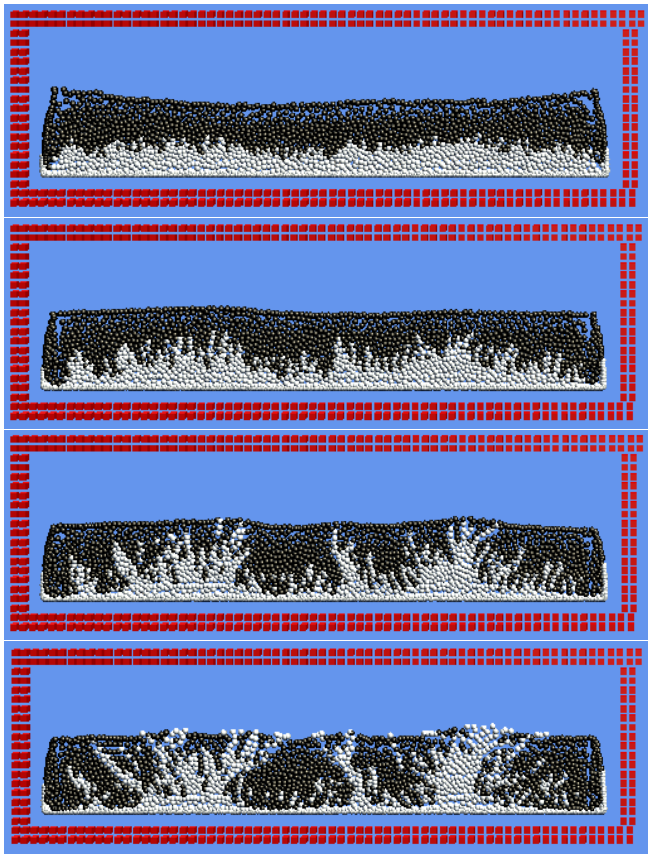

FIGURE 3.11: Mixing of the particles with a similar form to the development of salt diapirs.

particles of the lower density will reach the surface, and move unnaturally either to the left or right along the surface.

In reality diapirs are built through a process known as down-building [FK15]. Down-building is a geological process in which sedimentation occurs simultaneously to the development of the diapir. This would result in new layers of sediment particles (Black) being introduced at every time step. As a result, the diapir likely won't reach the surface, and instead, continue in its column-like shape beneath the surface.

3.3.1 Issues

The Tait equation for pressure was also implemented for the two-density system. However, this resulted in excessively large pressure forces causing an unstable surface to the particles. It also resulted in a "churning" of the particles, where particles on the boundary are pushed around the edges of the box. This caused many issues with the development of the diapirs.

In an attempt to have a thicker layer of sediment to allow for more time for the development of diapirs, the number of particles was also increased to upwards of 3000. However, it was discovered that the SPH program cannot support such a large number of particles for more than a few seconds, and the program becomes unstable. Given time constraints, there was no solution to this issue found.

3.4 Possible Improvements

There are a number of ways the program could be improved further. The first simplification to the project was to move from 3-D to 2-D. The SPH technique in three dimensions has a very similar formulation, with the main difference being that the algorithm must search for nearby particles in the ball of radius h, as opposed to the circle of radius h. As it can be imagined, this would be a much more computationally expensive process, and the number of particles would have to be increased greatly. As a result, the search algorithm would need to be modified to run even more efficiently. There are two main improvements that could be made. The first step is to take advantage of symmetry in the search algorithm. Namely, if a particle p_1 is a neighbour to particle p_2, then p_2 is a neighbour to p_1. As a result, the number of particles to be searched could be reduced further. The second method would be to implement a parallelized algorithm that would allow the program to compute the neighbourlist of various particles simultaneously.

As in [SP05], a force due to surface tension could be implemented for the free-flowing system. This would involve implementing a small spring-like force between particles.

As mentioned in section 3.3, implementation of a down-built diapir algorithm may produce more realistic results for the density contrast system. This could be achieved by having the program create a single layer of sediment particles at each time step, and have them deposit on the top of the free surface.

Another problem that was evident throughout the project was the implementation of the boundary force. In certain situations, particles near the boundary can be forced through the walls. This was handled by scaling the boundary force for the different cases. However, different boundary forces could be used that prevent particles from reaching the boundary instead of producing a large force away from the boundary.

As mentioned, there is certainly scope for further exploration into this project. Although the SPH technique developed provides a reasonable model for fluid dynamics, the algorithm could be modified further to provide a better model for the development of salt diapirs.

Bibliography

[Aue09] S. Auer. “Realtime particle-based fluid simulation”. MSc. Thesis. TU Munich, 2009.

[FK15] N. Fernandez and B. Kaus. “Pattern formation in 3-D numerical models of down-built diapirs initiated by a Rayleigh-Taylor Instability”. In: *Geophysical Journal International* 202 (2015).

[GM77] R.A. Gingold and J.J. Monaghan. “Smoothed particle hydrodynamics - Theory and application to non-spherical stars”. In: *Monthly Notices of the Royal Astronomical Society* 181 (1977).

[LL03] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics, a meshfree particle method*. World Scientific, 2003.

[Nei12] J. Neill. “Smoothed Particle Hydrodynamics”. MSc. Thesis. University College Cork, 2012.

[SP05] P. Beaudoin S. Clavet and P. Poulin. “Particle-based Viscoelastic Fluid Simulation”. In: *The Eurographics Association* (2005).

[SP08] B. Solenthaler and R. Pajarola. “Density Contrast SPH Interfaces”. In: *The Eurographics Association* (2008).