ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Test Suite for SPH Codes

Bachelor Thesis

Armin Riess

June 2, 2023

Supervised by Prof. Dr. R. Feldmann (UZH), Prof. Dr. S. Mishra (ETHZ)
Co-Advisors: Dr. D. Potter, T. Meier

Computational Science and Engineering, ETH Zürich

Contents

Contents **i**

1 Introduction **1**

2 Theory **3**
2.1 pkdgrav3 . . . 3
2.2 Smoothed Particle Hydrodynamics . . . 3
2.2.1 Overview . . . 4
2.2.2 Smoothing . . . 4
2.2.3 The Kernel . . . 4
2.2.4 Choice of Kernel . . . 5
2.2.5 Artificial Viscosity . . . 5
2.3 Tests . . . 6
2.3.1 Sod shock tube problem . . . 6
2.3.2 Sedov blast wave . . . 7

3 Initial Conditions **11**
3.1 Units . . . 11
3.2 Tipsy . . . 12
3.3 Glass-like Initial Conditions . . . 12
3.3.1 Random Distribution . . . 13
3.3.2 Grid . . . 13
3.3.3 Isentropes . . . 14
3.3.4 Dampening . . . 15
3.4 Initial Conditions for Tests . . . 18
3.4.1 ICs for Sod Shock Tube . . . 18
3.4.2 ICs for Sedov Blast Wave . . . 19

4 Results **21**
4.1 Running Simulations . . . 21

4.2 Problems 22
4.2.1 SPH Loop and Kick Factors 22
4.2.2 Incompatible Parameters 22
4.2.3 Time Step Calculation 23
4.3 Sod Shock Tube Test 24
4.4 Sedov Blast Wave Test 26
4.4.1 Ball Explosion 26
4.4.2 Single Particle Explosion 27

5 Conclusions **29**

Acknowledgments **31**

Bibliography **33**

Chapter 1

Introduction

Astrophysics and cosmology are two scientific fields where conducting experiments in a laboratory is impossible. The systems under study are astronomical in size, very far from Earth, and evolve on time scales that are orders of magnitudes larger than any human time scale or even the duration of a human life. Observations we make using telescopes on Earth and in space only give us a snapshot at a fixed time and have limited resolution. We also cannot influence the systems in space. Therefore, experiments have to be replaced by computer simulations.

Cosmological simulations are a powerful tool for studying the structure and evolution of the universe. N-body codes in particular are widely used to model the gravitational interactions in space, especially for dark matter. However, the dynamics of gas also play a crucial role in many astrophysical processes, for example during the formation of galaxies. Therefore, including hydrodynamics solvers in cosmological simulations is necessary to accurately model the universe.

These simulations have to be very complex and require immense computational power. The codes are run on supercomputers and have to be optimized to use the available hardware as efficiently as possible. This also makes developing and testing the codes very tedious. For this thesis, I created a suite of tests which can be used to determine the accuracy of hydrodynamics solvers.

The hydrodynamics code that was used for this thesis is integrated into the N-body code PKDGRAV3. PKDGRAV3 is a state of the art gravity solver that is capable of simulating trillions of particles. It was tested on some of the largest supercomputers in the world and delivered the fastest time-to-solution for large-scale cosmological N-body simulations at the time [8]. However, since only the hydrodynamics solver was tested in this thesis, the gravity solver was disabled.

The hydrodynamics solver examined in this project uses Smoothed Particle Hydrodynamics (SPH), a method widely used for computational fluid dynamics. I used it to simulate two problems, the Sod shock tube and the Sedov blast wave, both of which are standard tests to assess the accuracy of fluid dynamics codes. They both involve shock waves and discontinuities, which are challenging to capture with SPH. They also both have an analytical solution against which the results of the simulations can be compared to verify their accuracy.

The goal of this thesis was to create a suite of tests that can be used to determine the accuracy of hydrodynamics solvers, and to then run them using the SPH implementation in pkdgrav3.

Chapter 2

Theory

2.1 pkdgrav3

The original PKDGRAV published in 2001 [12] was developed to perform very high resolution cosmological N-body simulations of dark matter on hundreds of supercomputer nodes. It used a k-D tree, allowing it to scale with $O(N \log N)$ for large N, where N is the number of particles that are simulated.

The code is being maintained at the Institute for Computational Science of the University of Zurich. Since the original version it was developed further and improved. It was also used for other simulations, for example, the TreeSPH code GASOLINE [14] is an extension of PKDGRAV. The second version of PKDGRAV, PKDGRAV2, switched to using the fast multipole method (FMM). This method scales linearly with the number of particles, which led to a much better performance compared to the tree code that was used in the previous version.

Over the course of several years, the performance of PKDGRAV2 was gradually improved by adding GPU acceleration and other optimizations to run it on the CSCS supercomputer Piz Daint, which led to the latest version, PKDGRAV3. An implementation of MFM [1] and a new variant of SPH [6] were integrated into PKDGRAV3. This SPH implementation is the hydrodynamics solver which will be the focus of this thesis.

2.2 Smoothed Particle Hydrodynamics

This chapter will give a brief introduction of Smoothed Particle Hydrodynamics (SPH). It follows the descriptions given in [13], [5], and [6].

2.2.1 Overview

Smoothed-Particle Hydrodynamics (SPH) is a computational model that is commonly used to simulate fluids. It was first developed in 1977 for applications in astrophysics, but it also has applications in many other scientific fields [9]. SPH is a Lagrangian particle-based method, fluids are sampled with particles instead of with a mesh. This allows highly adaptive resolution, which makes it ideal to simulate phenomena that occur across many orders of magnitude. Since SPH uses particles, it can be integrated into particle-based N-body gravity codes for cosmological simulations.

2.2.2 Smoothing

SPH uses a kernel function W to smooth the interactions between particles. The kernel is a weighting function with a characteristic radius called the smoothing length h. For any fluid property q, such as density or pressure, the smoothed value q^s at any position $\vec{r}$ is defined as the integral of q at the particles around $\vec{r}$, weighted by the kernel function.

$$q^s(\vec{r}) = \int q(\vec{r}')W(|\vec{r} - \vec{r}'|, h)d\vec{r}' \tag{2.1}$$

This means that the fluid properties are evaluated using a weighted average over a small volume. The values of the particles are smoothed by the kernel function, and this smoothing gives the method the name “Smoothed Particle Hydrodynamics”.

2.2.3 The Kernel

The kernel function has to satisfy two properties: First, the kernel approaches the Dirac delta function as h approaches 0,

$$\lim_{h\to 0} W(|\vec{r} - \vec{r}'|, h) = \delta(|\vec{r} - \vec{r}'|), \tag{2.2}$$

and second, the kernel is normalized,

$$\int W(|\vec{r} - \vec{r}'|, h)d\vec{r}' = 1. \tag{2.3}$$

The integral over the kernel of particle i is discretized using a Riemann sum over the particles in the kernel:

$$q^s(\vec{r}_i) = q_i^s = \sum_j m_j \frac{q_j}{\rho_j} W(|\vec{r}_i - \vec{r}_j|, h_i), \tag{2.4}$$

where for a particle j, m_j is the mass, ρ_j the density, and q_j the value of some fluid property q. In every step of the SPH calculation, the smoothed density is calculated first,

$$\rho_i^s = \sum_j m_j \frac{\rho_j}{\rho_j} W(|\vec{r}_i - \vec{r}_j|, h_i) = \sum_j m_j W(|\vec{r}_i - \vec{r}_j|, h_i). \tag{2.5}$$

This ρ_i^s is then used in equation (2.4) instead of ρ_i to calculate all other necessary fluid properties.

The spacial derivative of the kernel can be used to estimate spacial derivatives of fluid properties [7]. This eliminates the need for finite differences or a grid.

$$\vec{\nabla} q^s(\vec{r}) = \sum_j m_j \frac{q_j}{\rho_j} \vec{\nabla} W(|\vec{r} - \vec{r}_j|, h_i) \tag{2.6}$$

2.2.4 Choice of Kernel

There are many functions that can be used as a kernel. Commonly used are the Gaussian or spline functions. Compact kernels with a finite interaction distance are better for computational performance since they only use the particles within the interaction distance. In contrast, when using the Gaussian kernel one has to sum over all particles, even those that are very far away and whose contribution is negligible, which is very inefficient.

The SPH implementation in PKDGRAV3 is a slightly adapted version of traditional SPH which doesn't use a fixed number of neighbours of the particles. By setting `iKernelType` in the parameter file to 0, 1, 2, or 3, the kernel type can be specified. The possible choices are the cubic spline M3 and the Wendland C2, C4, or C6 functions. M3 is most commonly used for SPH, and it was also the kernel I used for this project. It is defined as follows [6],

$$M_3(q) = \frac{4}{h^3 \pi} \begin{cases} 1 - 6q^2 + 6q^3, & 0 \leq q < \frac{1}{2} \\ -2(q-1)^3, & \frac{1}{2} \leq q < 1 \\ 0, & 1 \leq q \end{cases}, \tag{2.7}$$

where $q = \frac{r}{2h}$.

2.2.5 Artificial Viscosity

Most SPH implementation use an artificial viscosity to mimic the effects of viscosity, the internal friction present in fluids. It is needed to model dissipative processes and it also helps dealing with discontinuities and shear flows. Otherwise, these situations could lead to instabilities in the simulation.

Artificial viscosity introduces an additional term, which is defined in [6], into the discretized momentum equation. This term is defined for every pair of particles and it depends on the relative velocity, the average sound speed, average density, and average smoothing length of each pair. This results in an additional force between neighbouring particles to prevent them from passing through each other, making the fluid behave more like a continuum.

2.3 Tests

In this section I will discuss the theory behind the problems that were used to make the tests. The gas that was used is an ideal gas, which means it obeys the ideal gas law,

$$PV = Nk_BT, \tag{2.8}$$

and the polytropic equation of state,

$$P \propto \rho^\gamma, \tag{2.9}$$

where γ is the adiabatic index. For the Sod shock tube, γ is 1.4, and for the Sedov blast wave it is $\frac{5}{3}$.

2.3.1 Sod shock tube problem

The first test that was conducted is based on the Sod shock tube problem. The description below follows the one given in [3], where a derivation of the analytical solution for the general shock tube can also be found.

Shock tubes are a type of one-dimensional Riemann problem which are commonly used to test how well a fluid code can simulate discontinuities. Two fluids with different densities and pressures are placed next to each other, as if separated by a membrane.

At the start of the simulation at $t = 0$, the membrane between the fluids is removed. The fluid on the left side, where the pressure is higher, moves to the right side, where the pressure is lower. This leads to a discontinuity and a shock that travel to the right. Since the fluid moves from the left to the right and mass has to be conserved, there is also a rarefaction wave traveling to the left.

The domain can be divided into five regions, which are labeled in figure 2.1:

- The undisturbed fluid on the left (1)
- The rarefaction wave (2)
- Two regions with different densities but equal pressures, separated by the contact discontinuity (3 & 4)
- The undisturbed fluid on the right side of the shock (5)

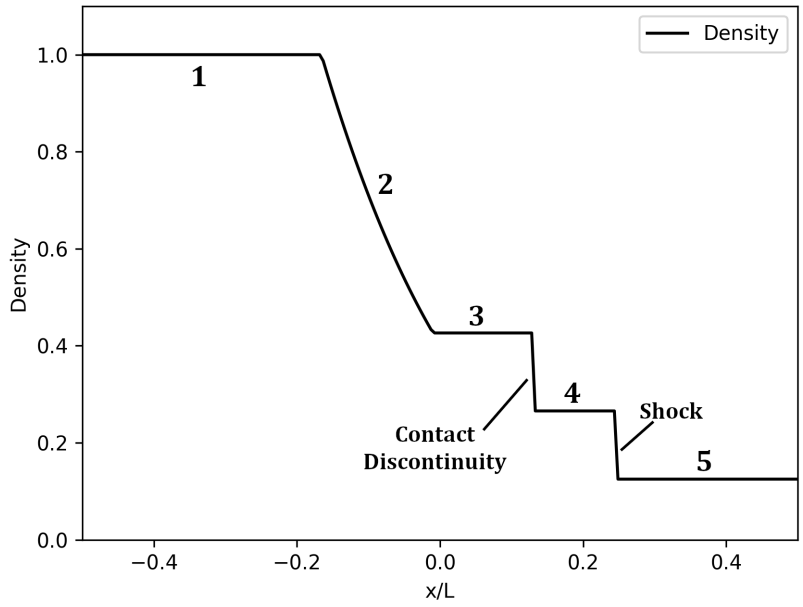

Figure 2.1: This is an example of an analytic density profile of a shock tube at $t = 0.5$. At $t = 0$, the interface between the two fluids was at $x = 0$. The 5 regions in the density, the contact discontinuity, and the shock are labeled.

The Sod shock tube is special variant of this problem with specific initial values for the density, pressure, and velocity in both regions:

$$\begin{pmatrix} \rho_L \\ P_L \\ v_L \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \rho_R \\ P_R \\ v_R \end{pmatrix} = \begin{pmatrix} 0.125 \\ 0.1 \\ 0 \end{pmatrix} \tag{2.10}$$

For this problem, the adiabatic index γ is equal to 1.4. These are the same parameters as in [11], the original paper by Gary A. Sod. They are also the initial values used for figure 2.1.

To calculate and plot the analytical solution the python script from [9] was used.

2.3.2 Sedov blast wave

The second test uses the Sedov blast wave problem. Initially, all particles are in equilibrium with a homogeneous density. At $t = 0$, a very large energy is induced in a very small volume. This is similar to what happens in a supernova explosion, where the energy is of the order of 10^{46} J. The gas near the explosion is blown outwards, a spherical blast wave forms at the

explosion that expands like a bubble. At first, the bubble grows very quickly, but then the surrounding gas slows it down.

We can derive the radius and the rate of expansion using dimensional analysis. The quantities we have are the radius r, the time t, the energy E of the blast, and the density ρ. The corresponding dimensions are:

$$[r] = \mathrm{L}, \quad [t] = \mathrm{T}, \quad [E] = \mathrm{M}\frac{\mathrm{L}^2}{\mathrm{T}^2}, \quad [\rho] = \frac{\mathrm{M}}{\mathrm{L}^3}. \tag{2.11}$$

We have $n = 4$ quantities and $k = 3$ dimensions, therefore, according to the Buckingham-Pi-Theorem, we can multiply some powers of these quantities and get $n - k = 1$ dimensionless parameter of order 1.

$$\Pi = r \cdot t^a \cdot E^b \cdot \rho^c \tag{2.12}$$

$$\Rightarrow 1 = \mathrm{L} \cdot \mathrm{T}^a \cdot \left(\mathrm{M}\frac{\mathrm{L}^2}{\mathrm{T}^2}\right)^b \cdot \left(\frac{\mathrm{M}}{\mathrm{L}^3}\right)^c. \tag{2.13}$$

By solving for the exponents, we get:

$$a = -\frac{2}{5}, \quad b = -\frac{1}{5}, \quad c = \frac{1}{5}. \tag{2.14}$$

Therefore,

$$\Pi = r \cdot t^{-\frac{2}{5}} \cdot E^{-\frac{1}{5}} \cdot \rho^{\frac{1}{5}}. \tag{2.15}$$

We get the solution by solving for r,

$$r = \Pi \cdot \left(\frac{E}{\rho}\right)^{\frac{1}{5}} t^{\frac{2}{5}}. \tag{2.16}$$

The shock speed is the derivative of r with respect to time t,

$$\frac{dr}{dt} = \Pi \cdot \frac{2}{5}\left(\frac{E}{\rho}\right)^{\frac{1}{5}} t^{-\frac{3}{5}}. \tag{2.17}$$

The full density and pressure profile is given by the self-similar Sedov solution. From the one-dimensional Euler equations, a system of three differential equations can be derived and solved numerically. The full derivation can be found in [10]. The analytical density of the Sedov blast wave used in this thesis is the one used in [1].

Figures 2.2 and 2.3 show an example of a Sedov blast wave. Figure 2.2 was created by first sampling the domain with points and using the analytical density shown in figure 2.3 to color them.

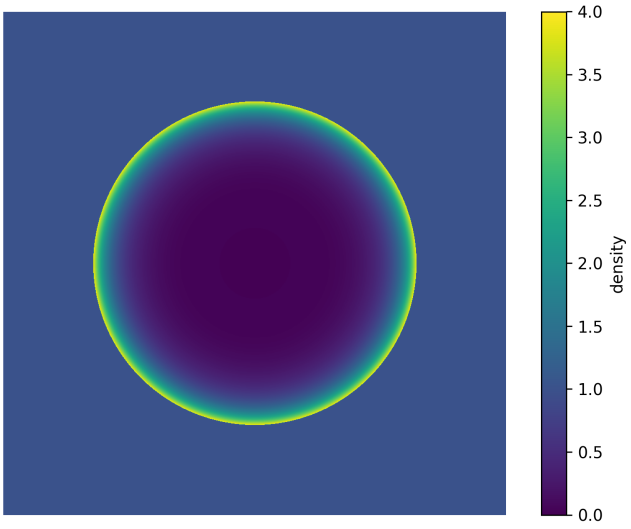

Figure 2.2: An example of an analytic 2-dimensional Sedov blast wave.

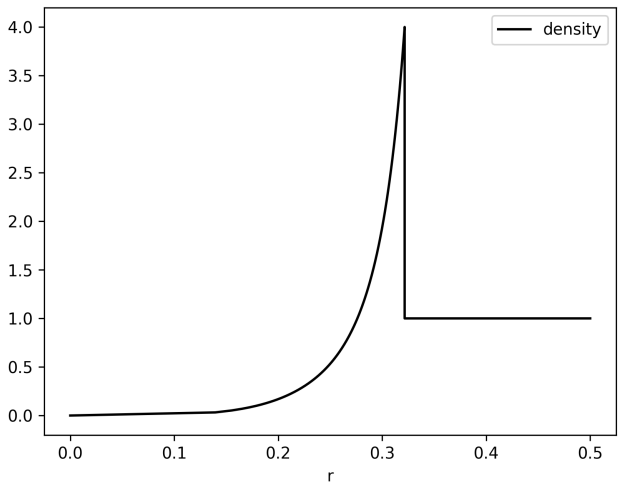

Figure 2.3: The analytic density profile over the radius for figure 2.2

Chapter 3

Initial Conditions

3.1 Units

Like many N-body simulations, PKDGRAV3 uses its own system of units in which the gravitational constant G is 1. The code units of mass and length are also chosen such that the total mass and the length of the domain are 1. Setting these parameters to 1 makes the calculation of gravity slightly more efficient and the values of variables in the code are easier to work with.

However, to make sense of the results of the simulation, we have to choose some physical units that correspond to the code units. In the parameter file, one can choose how much a code mass unit should be in solar masses and how much a code length unit in kiloparsec. This way, one can set the length of the domain and the total mass. Then the unit of time can be calculated. In the following, I will show how the units that were used for the Sod shock tube test were derived. The code units for length, mass, time, velocity, and energy are denoted by L, M, T, V, and E, respectively.

In some of the tests that were conducted for this thesis, a gas cloud of non-ionized hydrogen with an average density of 100 atoms per cm^3 and a temperature of 10K was simulated. This corresponds to an interstellar molecular gas cloud [2]. The length of the box is set to be 20 pc.

$$\mathrm{L} = 20\ \mathrm{pc} = 6.17 \cdot 10^{17}\ \mathrm{m} \tag{3.1}$$

$$\rho = \frac{\mathrm{M}}{\mathrm{L}^3} = 100\ \frac{\mathrm{m_H}}{\mathrm{cm}^3} = 1.66 \cdot 10^{-19}\ \frac{\mathrm{kg}}{\mathrm{m}^3} = 2.45\ \frac{\mathrm{M_\odot}}{\mathrm{pc}^3} \tag{3.2}$$

$$\Rightarrow \mathrm{M} = 1.96 \cdot 10^4\ \mathrm{M_\odot} = 3.90 \cdot 10^{34}\ \mathrm{kg} \tag{3.3}$$

Using these quantities and $G = 1$, other units can be derived:

$$G = 6.67 \cdot 10^{-11} \left(\frac{\mathrm{m}}{\mathrm{L}}\right)^3 \left(\frac{\mathrm{s}}{\mathrm{T}}\right)^{-2} \left(\frac{\mathrm{kg}}{\mathrm{M}}\right)^{-1} = 1 \tag{3.4}$$

$$\Rightarrow \mathrm{T} = \sqrt{\frac{G}{6.67 \cdot 10^{-11}} \left(\frac{\mathrm{L}}{\mathrm{m}}\right)^3 \frac{\mathrm{kg}}{\mathrm{M}}}\ \mathrm{s}$$

$$= 3.00 \cdot 10^{14}\ \mathrm{s} = 9.51\ \mathrm{Myr} \tag{3.5}$$

$$\mathrm{V} = \frac{\mathrm{L}}{\mathrm{T}} = 2.06 \cdot 10^3\ \frac{\mathrm{m}}{\mathrm{s}} \tag{3.6}$$

$$\mathrm{E} = \mathrm{MV}^2 = 1.66 \cdot 10^{41}\ \mathrm{J} \tag{3.7}$$

Temperature does not need to be converted to SI units as the code unit for temperature is Kelvin.

In the parameter file for pkdgrav3 there are two parameters, `dKpcUnit` and `dMsolUnit`, which define what the code units correspond to in physical units. `dKpcUnit` is how long a code length unit is in kiloparsecs and `dMsolUnit` is how much a code mass unit is in solar masses. According to equation (3.1), the code length unit is equal to 20 parsecs, therefore a kiloparsec is equal to 0.02 code unit lengths. Similarly, according to (3.3), the code unit of mass is equal to $1.96 \cdot 10^4$ solar masses.

```
dKpcUnit = 2e-2
```
(3.8)

```
dMsolUnit = 1.96e4
```
(3.9)

3.2 Tipsy

The files that pkdgrav3 reads and writes are in tipsy-format. These files contain the important properties of the particles for every step. There are three different types of particles that can be simulated: gas, dark matter, and stars, but since the focus of this project was on gas dynamics, only gas particles were used. The header of a file contains the time, number of particles in total and for each type, and the number of dimensions. The files can be stored in little endian or big endian, the default in pkdgrav3 is big endian. In this project, all files were stored in little endian since that is what Euler uses, which made it much easier to write to the binary files. pkdgrav3 can also use other file types such as hdf5, but only tipsy was used for this project.

3.3 Glass-like Initial Conditions

To create initial conditions for the tests we first need a glass-like distribution of particles, which means that the particles are randomly distributed, in

equilibrium, and at rest. The fluid should also be isotropic, otherwise the speed of a shock might depend on its direction.

3.3.1 Random Distribution

A simple approach is to give all the particles a random position from a uniform distribution across the whole domain. This is very easy to do and was good enough for some initial tests, but there are also a couple of problems with this approach. First, the particles are not in equilibrium. Second, there is a lot of noise, as shown in figure 3.1. If, by chance, some particles are closer together while others are further apart, their densities are different. For the tests, we want to compare density distributions, so it is important to minimize the noise such that all particles have the same density.

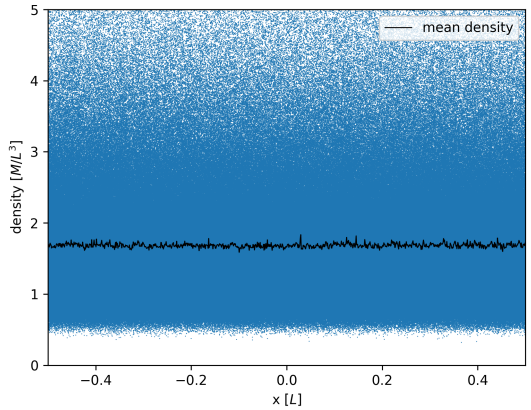

Figure 3.1: When the particles are distributed randomly in the domain, the density fluctuates a lot. This plot shows the densities of all particles over their x-coordinate, and the mean density. The mean density should be around 1, but since the density of a particle can be arbitrarily high but not lower than 0, the mean density of all particles will usually be larger than 1.

3.3.2 Grid

A different approach is to place the particles on a grid. This way, all particles have the same initial density. However, a problem with using a grid is that the distribution is not isotropic. The distances between particles along the grid lines is shorter than along the diagonals, which could change the speed of a shock depending on in which direction it is travelling.

To fix this, we can place the particles on a grid, and then add some randomness to the positions, for example by drawing a random position from a uniform distribution around a grid node. Then, we simulate the particles to

let them move around and settle into an equilibrium. This is called relaxing. Figure 3.2 shows the noise in the density before and after relaxing.

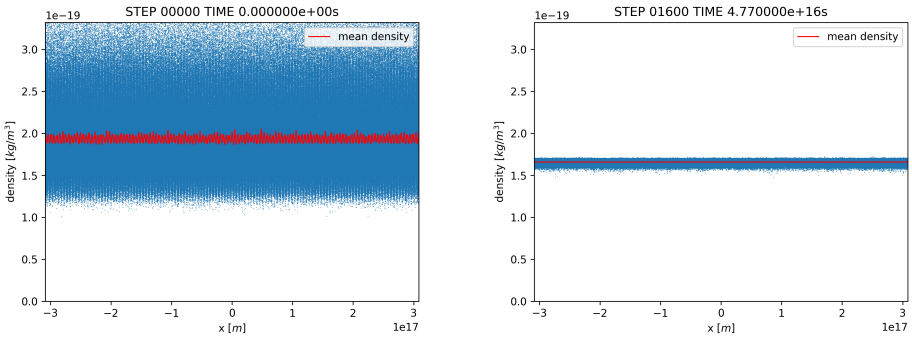

Figure 3.2: The particles are placed on a grid with small perturbations in the position, then they are relaxed. Before relaxing, the noise in the density is much higher. After relaxing the particles for 1600 steps, the density is much closer to uniform. In the plot on the left the structure in the particle distribution due to the grid is also visible.

A similar method would be to place the particles on a grid with no randomness, and set the velocity randomly instead of to 0. In that case, the density is perfectly uniform in the beginning and due to the non-zero velocities the particles still move around so the initial grid structure disappears. I ran a quick test, but I found that this not easier than the method I used. This will be discussed in section 3.3.4.

3.3.3 Isentropes

The relaxation is mostly an adiabatic process, and as such it is characterized by the polytropic process equation,

$$PV^{\gamma} = \text{const}, \tag{3.10}$$

with pressure P, volume V and adiabatic index γ. We can also transform the ideal gas law,

$$PV = Nk_BT \Rightarrow PVT^{-1} = \text{const}, \tag{3.11}$$

where N is the number of moles, k_B the Boltzmann constant, and T the temperature. Dividing equation (3.10) by equation (3.11) gives us

$$V^{\gamma-1}T = \text{const}. \tag{3.12}$$

Finally, since the mass of a particle is constant, $\rho \propto V^{-1}$, and thus,

$$T\rho^{1-\gamma} = \text{const}. \tag{3.13}$$

This relation characterises the isentrope of a particle. As long as the relaxation is an adiabatic process, the term $T\rho^{1-\gamma}$ will be constant for each particle and they will stay on their isentrope.

At the end of the relaxation, the particles should all have approximately the same density. The same is true for the pressure, because as long as there are pressure gradients the particles are not in equilibrium. Since the pressure is a function of density and temperature,

$$P = \frac{k_B}{m_H}\rho T, \tag{3.14}$$

this means that the temperature should also be close to uniform. Therefore, the particles should all be on the same isentrope at the end of the relaxation, or at least close to it.

Because the particles have different densities at the beginning, we have to adjust the temperatures of the particles according to their densities to place them on the same isentrope. This can be done by solving equation (3.13) for the temperature. By choosing the target density ρ_* and target temperature T_*, we can calculate the temperature of any particle i.

$$T_*\rho_*^{1-\gamma} = T_i\rho_i^{1-\gamma} \Rightarrow T_i = T_*\left(\frac{\rho_*}{\rho_i}\right)^{1-\gamma} \tag{3.15}$$

Placing the particles on the same isentrope at the start of the relaxation run only works well when the density is already close to uniform. If this is not the case, small shocks can form at regions with higher density. At shocks, the polytropic process equation (3.10) does not hold and the temperature of the particles rises. After the shock has disappeared, the particles are no longer on the same isentrope and thus the results of the relaxation are worse.

It would be best to do this process iteratively by placing the particles back on the same isentrope every few steps during the relaxation. This would be a possible extension to this project.

3.3.4 Dampening

If the density in a region is higher than the average in the whole box, this introduces an oscillation. The particles move out of that high density region until the density is too low, and then the density will increase again to what it was at the beginning. In the initial conditions of the tests the particles should be at rest, i.e., have as little kinetic energy as possible. To achieve this, one can dampen the velocities during the relaxation by scaling them down at every substep.

The dampening used by PKDGRAV3 scales the velocities at each step according to

$$\frac{d\vec{v}}{dt} = -a \cdot \vec{v} \Rightarrow \vec{v}' = e^{-a\cdot\Delta t} \cdot \vec{v}, \tag{3.16}$$

where a is the dampening parameter and Δt is the size of the timestep of the current particle.

A good choice of the dampening parameter is essential. If the dampening is too weak, it will take a lot of steps until the particles are at rest. If the dampening is too strong, the kinetic energy is reduced too quickly, before the particles have time to settle into an equilibrium. A good dampening parameter can be found by testing different values and comparing the results.

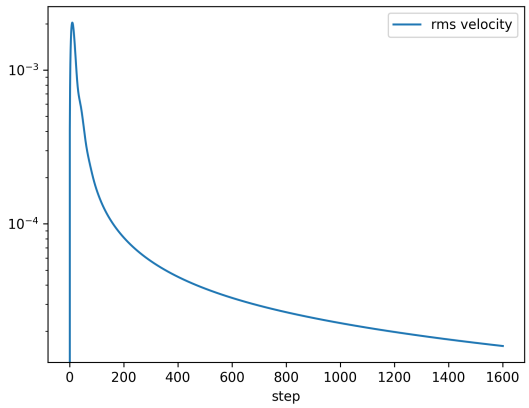

Figure 3.3: This plot is from a relaxation simulation of 128^3 particles for 1600 steps with a dampening of 0.05. At first, the velocities are 0, then due to the differences in densities the particles are accelerated, and the dampening gradually decelerates them again.

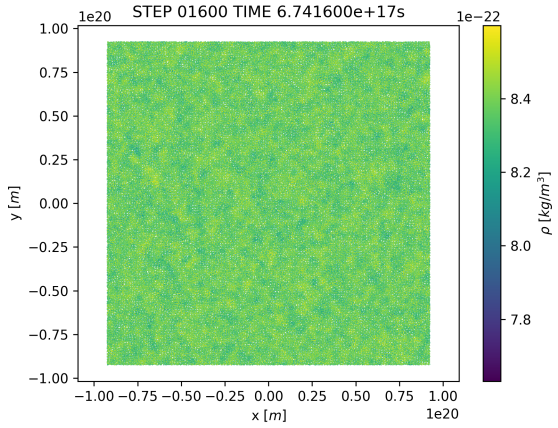

Figure 3.4: This is a slice through the particles along the xy-plane after relaxing for 1600 steps. The structure of the grid is no longer noticeable.

Figure 3.3 is from a relaxation simulation of 128^3 particles. It shows the RMS

velocity in code units for each step. In the initial conditions the velocity is zero. Because the positions of the particles are random, the pressure is not uniform and thus the particles are accelerated, which leads to high velocities. Due to the dampening, the velocities decrease over time. After 1600 steps, the particles are not perfectly relaxed, but the velocities are sufficiently small.

Figure 3.4 shows that after relaxing the particles have moved far enough from the grid so there is no longer any structure visible. We can assume that the distribution of the particles is random, which is one of the requirements for a glass-like distribution.

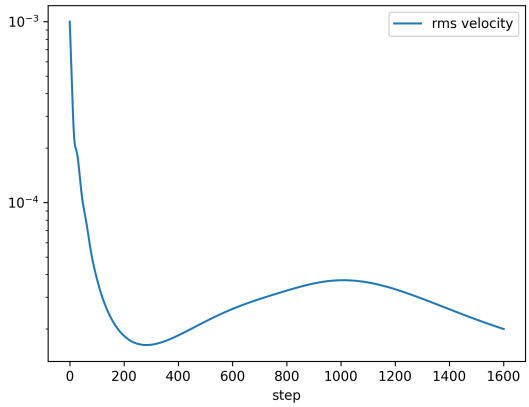

Figure 3.5: This plot is from the quick test that was mentioned at the end of section 3.3.2. It shows the RMS velocity during the relaxation. In the beginning, the particles were placed on a grid with random velocities.

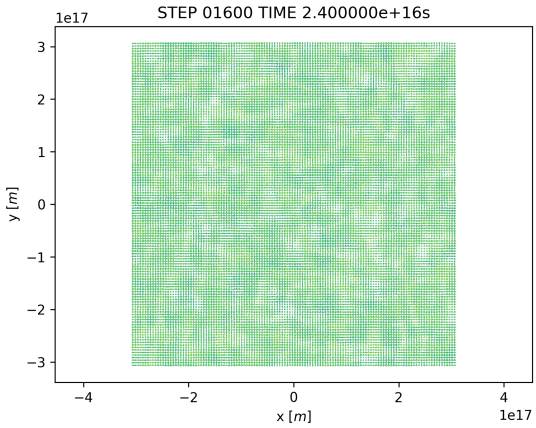

Figure 3.6: This slice along the xy-plane from $z = -0.1$ to $z = 0.1$ is the result of the relaxation shown in figure 3.5. The grid is still visible, so this particle distribution is not a good glass.

Figure 3.5 is from the quick test that was explained at the end of section 3.3.2. It shows the RMS velocity for the particles that were placed on a grid with random velocities. This time, the RMS velocity does not gradually become smaller, instead it oscillates. Also, the grid structure is still visible, as shown in figure 3.6, indicating that either the initial velocity was too small or the dampening too strong. However, this was just a quick test, by experimenting with different initial velocities and dampening parameters a better particle distribution could be achieved.

3.4 Initial Conditions for Tests

3.4.1 ICs for Sod Shock Tube

For the Sod shock tube, we need two regions with different densities and pressures. There are different ways to change the density, either by changing the amount of particles or the mass per particle. The method chosen for this project was to vary the number of particles, with every particle having the same mass.

First, two boxes with 64^3 and 128^3 particles were relaxed. Then, one half of both boxes were used and placed together in the domain. The distribution with 128^3 particles was used for the region with high density on the left ($x < 0$), and the distribution with 64^3 particles was used for the region with low density on the right ($0 < x$). Then, the mass was adjusted such that all particles have the same mass and the total mass is still equal to 1 in code units. The temperature was changed as well, first to place the particles again on the same isentrope and then to change the pressure in the right region such that the pressure ratio of the two regions is 1:10.

$$\begin{aligned} P_L &= \frac{k_B}{m_H} T_L \rho_L \\ P_R &= \frac{k_B}{m_H} T_R \rho_R = \frac{1}{10} P_L = \frac{1}{10} \frac{k_B}{m_H} T_L \rho_L = \frac{1}{10} \frac{k_B}{m_H} T_L \cdot 8\rho_R \\ \Rightarrow T_R &= \frac{8}{10} T_L \end{aligned} \tag{3.17}$$

However, even though the number of particles is quite high, the resolution of the shock is quite poor, which will be shown in section 4.3. The Sod shock tube is essentially a one-dimensional problem, which means that having a lot of particles in the y and z directions does not improve the resolution. To achieve a higher resolution, I moved from a cubic box to a longer tube. For the left region, a $256 \times 32 \times 32$ grid was used to place the particles, and for the right region a $128 \times 16 \times 16$ grid. The number of particles in the left region is 8 times higher than in the right region, which means that also the density is 8 times higher, which corresponds to the density ratio in the Sod shock tube.

To get the correct pressures in the initial conditions, I again scaled the temperature in the right region by $\frac{8}{10}$. Then, the temperature is 10K in the left region and 8K in the right region.

After relaxing, the two boxes were placed next to each other and scaled such that the total length of the tube is 1 in code units. This time, all particles from the relaxation were used, not just half of them. Now, the number of particles along the x axis was 4 times higher, and the total number of particles was 8 times lower compared to initial conditions with the cubic box.

In short, $256 \cdot 32 \cdot 32 + 128 \cdot 16 \cdot 16 = 294912$ particles were used in total. Both regions of the Sod shock tube were relaxed separately for 1600 steps with a time step of 0.05 and a dampening of 0.05, then they were combined. In the final test setup, the left region had a density of $\frac{16}{9}\frac{\mathrm{M}}{\mathrm{L}^3} = 2.95 \cdot 10^{-19}\frac{\mathrm{kg}}{\mathrm{m}^3}$, a temperature of 10K, and a pressure of $\frac{k_B}{m_H} \cdot 2.95 \cdot 10^{-19}\frac{\mathrm{kg}}{\mathrm{m}^3} \cdot 10\mathrm{K} = 2.45 \cdot 10^{-14}\mathrm{Pa}$. The right region had a density of $\frac{2}{9}\frac{\mathrm{M}}{\mathrm{L}^3} = 3.69 \cdot 10^{-20}\frac{\mathrm{kg}}{\mathrm{m}^3}$, a temperature of 8K, and a pressure of $\frac{k_B}{m_H} \cdot 3.69 \cdot 10^{-20}\frac{\mathrm{kg}}{\mathrm{m}^3} \cdot 8\mathrm{K} = 2.45 \cdot 10^{-15}\mathrm{Pa}$.

3.4.2 ICs for Sedov Blast Wave

The Sedov blast wave test uses a cubic box and 128^3 particles. I experimented with different methods to add the energy of the blast to the middle of the domain, first by changing the temperature of the particles in a ball with a radius of one 1/50 of the length of the box, and then by adjusting the temperature of a single particle. Using multiple particles is easier to resolve for the simulation, because the difference in internal energy between the particles in the ball and the ones outside is smaller.

For this test, the code units were not the ones derived in chapter 3.1, but the ones used for the Sedov blast wave in [4], so the results could be compared. Also, the energy of the blast has a lower value in the new code energy unit, which means the blast is easier to simulate. The box has a side length of 6 kpc and a density of $n = 0.5\ \mathrm{cm}^{-3}$. The particles have a temperature of 10K, a mass of $2.51 \cdot 10^{33}\ \mathrm{kg} = 1.26 \cdot 10^3\ \mathrm{M}_\odot$, and γ is $\frac{5}{3}$. The energy of the explosion is $6.78 \cdot 10^{46}$ J.

```
dKpcUnit = 6                    (3.18)
dMsolUnit = 2.65e9              (3.19)
```

The energy of the blast depends on the temperature in the ball according to

$$E = \frac{3}{2}\frac{m_b}{m_\mathrm{H}}k_B T, \tag{3.20}$$

where m_H is the mass of a hydrogen atom, k_B the Boltzmann constant, and m_b is the mass and T the temperature of the particles where the energy is

induced. We can use this equation to calculate what the temperature of the particles should be. For the explosion started by a single particle, this is

$$T = \frac{2}{3}\frac{m_{\mathrm{H}}}{m_b}\frac{E}{k_B} = \frac{2 \cdot 1.66 \cdot 10^{-27}\ \mathrm{kg} \cdot 6.78 \cdot 10^{46}\ \mathrm{J}}{3 \cdot 2.51 \cdot 10^{33}\ \mathrm{kg} \cdot 1.38 \cdot 10^{-23}\ \frac{\mathrm{J}}{\mathrm{K}}} = 2.17 \cdot 10^{9}\ \mathrm{K}, \tag{3.21}$$

and for the ball with a radius of 0.02 code length units which happened to contain 71 particles,

$$T = \frac{2}{3}\frac{m_{\mathrm{H}}}{m_b}\frac{E}{k_B} = \frac{2 \cdot 1.66 \cdot 10^{-27}\ \mathrm{kg} \cdot 6.78 \cdot 10^{46}\ \mathrm{J}}{3 \cdot 71 \cdot 2.51 \cdot 10^{33}\ \mathrm{kg} \cdot 1.38 \cdot 10^{-23}\ \frac{\mathrm{J}}{\mathrm{K}}} = 3.06 \cdot 10^{7}\ \mathrm{K}. \tag{3.22}$$

In short, for the Sedov blast test I used $128^3 = 2097152$ particles with a density of $\frac{\mathrm{M}}{\mathrm{L}^3} = 8.32 \cdot 10^{-22}\frac{\mathrm{kg}}{\mathrm{m}}$ and a temperature of 10K. The particles were relaxed for 1600 steps with a time step of 0.1 and a dampening of 0.01. The energy of the blast was $6.78 \cdot 10^{46}$J, and it was either added by setting the temperature of a single particle to $2.17 \cdot 10^{9}$ K or by setting the temperature of the central 67 particles of the box to $3.06 \cdot 10^{7}$ K. The blast was simulated for 300 steps with a time step of 0.0005.

Chapter 4

Results

In this chapter, I will show how I ran the simulations, what went wrong, and what the results were. The programs I wrote for this project to generate the ICs and to plot the results, along with the final parameter files and initial conditions of the simulations can be found on this repository: https://github.com/Artisorak/Bachelor-Thesis.

4.1 Running Simulations

Most of the simulations for this project were run on Euler, the computing cluster of ETH Zurich. The tests are not very computationally expensive, but running them on Euler made it easier to load the necessary libraries and to run multiple simulations at the same time. I used 16 to 24 cores of an AMD EPYC 7742 64-core processor. PKDGRAV3 is also optimized for GPU acceleration, but I did not have access to GPUs on Euler. Slurm was used to submit the jobs on Euler.

Due to the way PKDGRAV3 decides how to use the available resources, the number of nodes, tasks, and cores has to be specified very precisely. For example, for running on one node with 16 cores, the options have to be specified as follows:

```
--nodes=1
--ntasks-per-node=1
--cpus-per-task=16
```

It is also important to launch the executable using `srun`, otherwise the code may not run in parallel.

In the following sections I will first explain some of the issues that occurred during this project and then show the results of the simulations for each of the tests and how they compare to the analytical solutions.

4.2 Problems

While working on this project, I ran into a few problems. The tests used in this thesis ignore gravitational effects. By setting the parameter `bDoGravity` to 1 or 0, one can turn gravity on or off. However, turning off gravity caused several issues. pkdgrav3 had never been tested without gravity before and a lot of other aspects of the code relied on the gravity calculation.

All of these problems have since been fixed.

4.2.1 SPH Loop and Kick Factors

First, the SPH force calculation used the loop that does the gravity calculation. When gravity was turned off the loop was no longer executed, meaning that the forces were not calculated and the particles did not move. After this was fixed another bug was discovered. During the density calculation the kick factors of the particles were stored in memory. However, in the force calculation the kick factors were never initialized. However, because the memory was always allocated at the same location and it still contained the values of the previous step, the simulation always reused the kick factors from the previous step, and so it ran nonetheless.

4.2.2 Incompatible Parameters

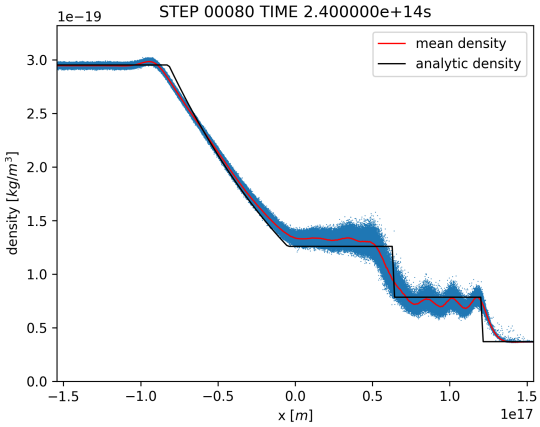

Figure 4.1: First test of the Sod shock tube. The result is incorrect, the density is very noisy and has large fluctuations. This was caused by two incompatible parameters.

The parameter file I used at first had both `bGasOnTheFlyPrediction` and `bGasConsistentPrediction` turned on. These are two methods used during the SPH force calculation to estimate the velocities of the gas particles, but they are incompatible. This led to strange behaviour in the Sod shock

tube, which is shown in figure 4.1. The density profile has large fluctuations, which were caused by the incompatible gas prediction methods. `bGasOnTheFlyPrediction` was then turned off, and now there are checks to make sure that there are no incompatible parameters set.

4.2.3 Time Step Calculation

Another problem became evident after running the Sedov blast test. During the first step, some particles traveled further than the side length of the box. PKDGRAV3 enforces the periodic boundary conditions by checking if the particles are outside of the box and adding or subtracting one box length from the corresponding coordinate. If a particle moves more than a box length in a time step it will still not be in the box after the periodic boundary conditions are enforced, which is a sign that something has gone very wrong.

The problem can be seen in this test with a much smaller maximum time step, so the particles would not move outside of the box.

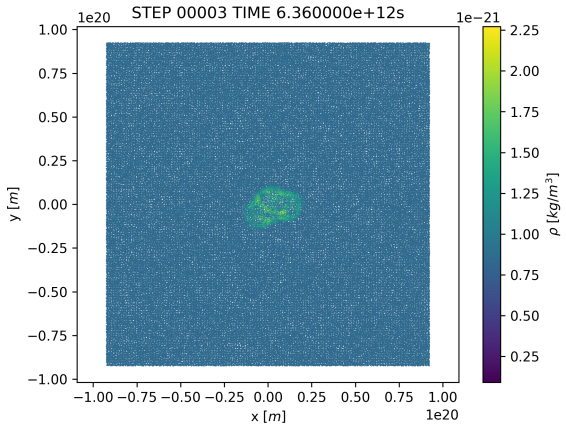

Figure 4.2: One of the first runs of the Sedov blast started by a single particle after 3 steps. The blast is not spherical, it looks like there are several blasts starting at different particles. This was caused by an incorrect time step calculation.

Already after the third step the blast is no longer spherical, as shown in figure 4.2. It looks like the explosion is starting at more than one location.

There was an error in the time step calculation that was introduced when the force calculation loop of SPH was changed so gravity could be turned off. Both the SPH and gravity forces give a maximum time step, and the actual time step is the minimum of the two. For the SPH interaction between particles i and j, the maximum SPH time step is calculated using the smoothing length, the Courant parameter, and some parameters from the artificial

viscosity. The gravity time step condition uses the particle softening and the magnitude of the acceleration.

Therefore, acceleration is only taken into account in the gravity time step condition. Since gravity was turned off, only the SPH time step was used and the acceleration had no impact on the time step. Due to the large pressure gradient at the explosion, the velocities and thus the displacement became very large, which resulted in unphysical behaviour. The time step calculation is explained in more detail in [6].

Because the time step was too large, the displacements of some of the particles was too high, which led to the incorrect results.

4.3 Sod Shock Tube Test

As mentioned in chapter 3.4.1, I experimented with different box dimensions for the Sod shock tube test. At first, I used a cubic box which was divided into two halves with different numbers of particles, one with $\frac{128^3}{2}$ particles and the other with $\frac{64^3}{2}$ particles.

However, the high number of particles hides the fact that the resolution in the direction of the shock is quite small. Since the Sod shock tube is essentially a one-dimensional problem, the particles in y and z direction do not add to the resolution. Also, since the boundary conditions are periodic, there are two shocks traveling in opposite directions, one starting at $x = 0.0$ and the other at $x = 0.5$, as shown in figure 4.3. These two shocks collide at $x = 0.25$.

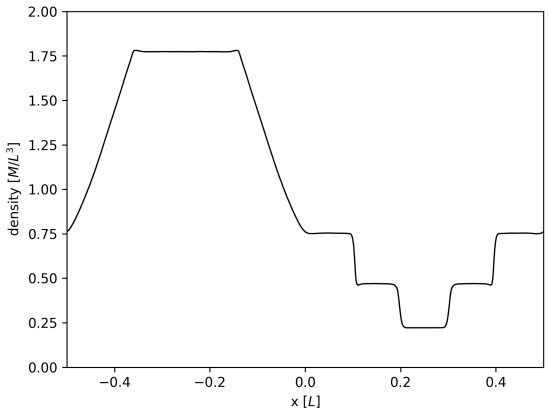

Figure 4.3: Since the boundary conditions are periodic, there are two shocks traveling in opposite directions. This plot shows the density across the whole box. Only the particles between $x = -0.25$ and $x = 0.25$ can be used, so only half of the box.

Once the two shocks meet, the results of the simulation cannot match the analytical solution since in the analytical solution there is only one shock in an infinitely long box. Therefore, only the particles between $x = -0.25$ and $x = 0.25$ are relevant for this test, which is half of the particles. In the right region, the one with lower density, there are $\frac{64^3}{2} = 131072$ particles, but only 16 particles along the x-axis, which is the direction in which the shock travels. This means that the simulation uses a high number of particles, but the resolution of the shock is very poor.

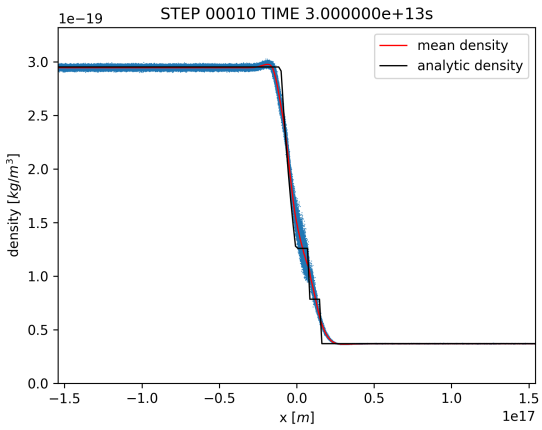

Figure 4.4: With the cubic box, the resolution in the direction in which the shock travels is too low. After 10 steps, the shock wave has barely moved by one smoothing length, and the distinctive steps in the density are not present.

Even after 10 steps, the shock has barely moved by one smoothing length and the steps in the density are not visible at all. The resolution is too low and there is no chance to capture the discontinuities accurately.

To fix this, a very long box was used instead, resulting in a resolution of 128 particles for the high density region and 64 particles in the low density region. This means that the resolution along the x axis is 4 times higher, and the total number of particles is 8 times lower compared the cubic box.

This time, the results of the simulation match the analytical solution very well. The discontinuities are much sharper, which shows that the resolution is better. The density in regions 4 and 5 is correct and any fluctuations are gone. There is a spike in the pressure plot at the contact discontinuity, this is called a “pressure blip” and it is expected with SPH.

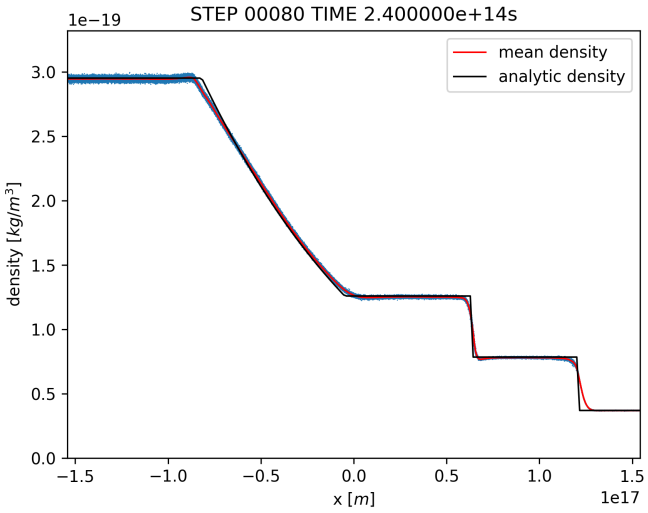

Figure 4.5: Final results of the Sod shock tube test: Density compared to the analytical solution.

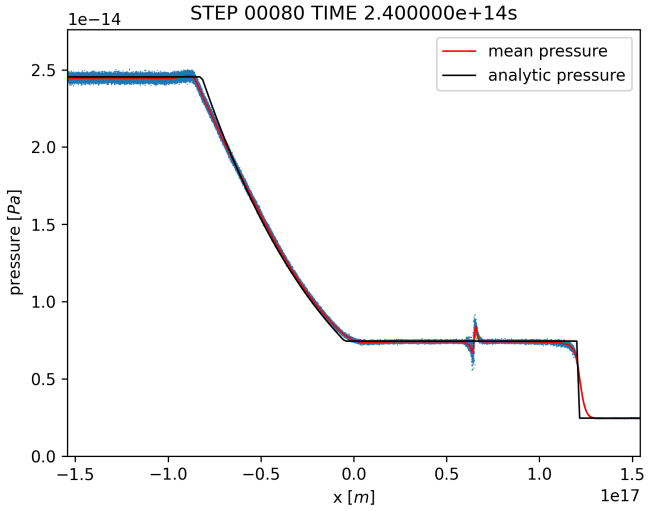

Figure 4.6: Final results of the Sod shock tube test: Pressure compared to the analytical solution.

4.4 Sedov Blast Wave Test

4.4.1 Ball Explosion

The Sedov blast wave test worked best when injecting the energy into a ball. This can be seen in figures 4.7 and 4.8. The radius of the blast and the density after the shock match the analytical solution perfectly. The peak is not as high and not as sharp, but this is an expected effect of the smoothing in SPH.

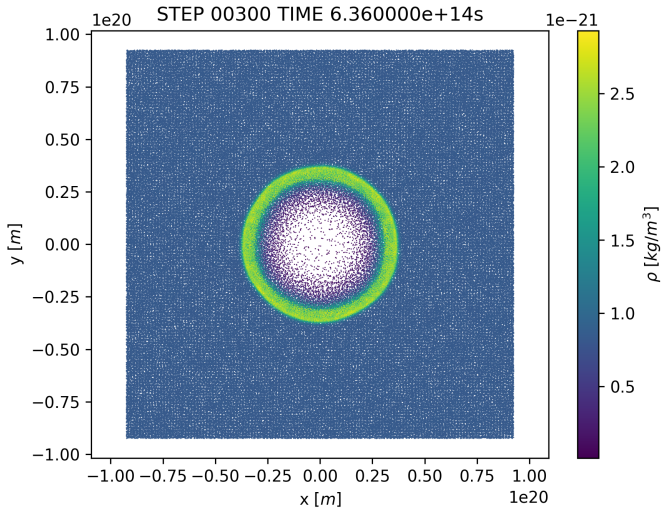

Figure 4.7: A slice in the xy-plane of the blast started by a ball after 300 steps, colored by density.

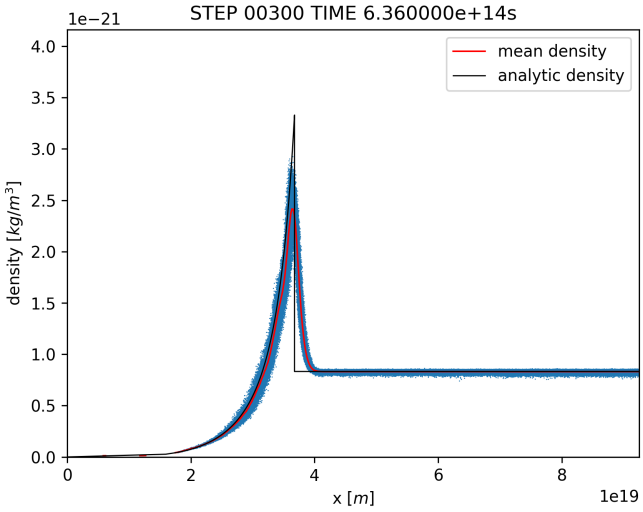

Figure 4.8: Density profile of the Sedov blast wave started by a ball after 300 steps.

4.4.2 Single Particle Explosion

After the problem with the timestepping was fixed, the explosion started by giving the energy to a single particle worked as well. The densities of the particles are slightly more spread out, which can be seen in figure 4.10.

The Sedov blast wave test is much harder to model correctly in a simulation compared to the Sod shock tube test. However, the results show that the SPH

implementation in pkdgrav3 is capable of handling the enormous energy and capturing the blast perfectly.

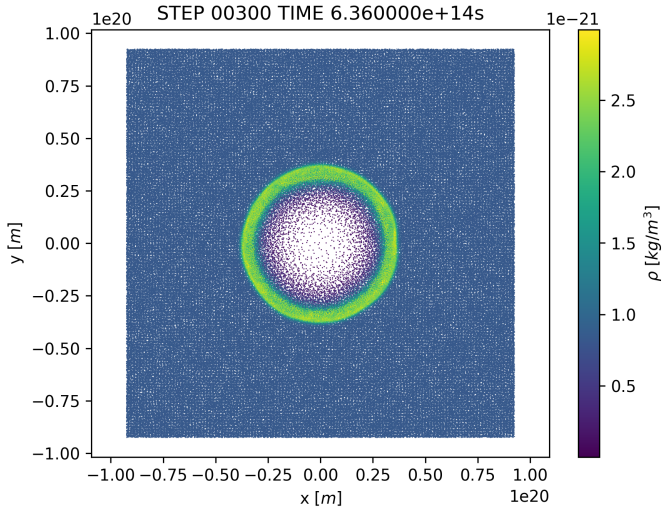

Figure 4.9: A slice in the xy-plane of the blast started by a single particle after 300 steps, colored by density.

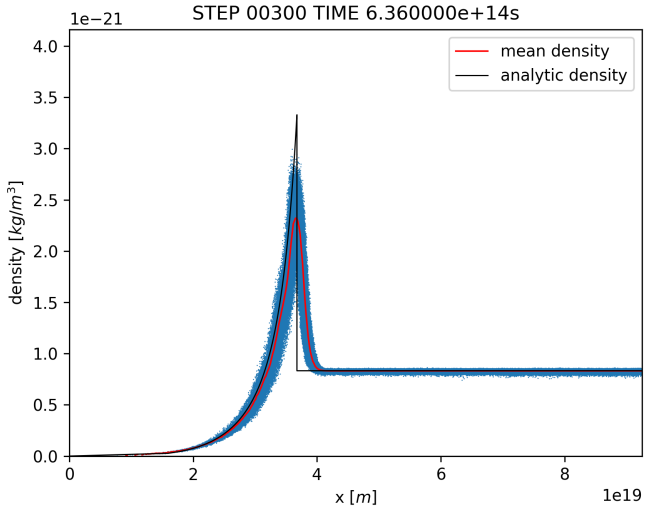

Figure 4.10: Density profile of the Sedov blast wave started by a single particle after 300 steps.

Chapter 5

Conclusions

The goal of this bachelor thesis was to create tests for SPH codes, and then to run them with the SPH implementation in PKDGRAV3 to see if it is working correctly. To do this, two problems that are often used to test fluiddynamics simulations were used, the Sod shock tube and the Sedov blast wave.

To create the initial conditions for both tests, a glass-like distribution of particles was needed. Different methods to place the particles in the box were tried, first using random positions and then using a grid with some random perturbations in the positions. To make sure that the particles can go towards uniform density they were placed on the same isentrope. Then, the particles were relaxed for several hundred steps to let them move towards equilibrium. Since the particles should be close to stationary at the beginning of the tests, the velocity of the particles was dampened during the relaxation.

The relaxed particles were then altered and combined to create the initial conditions for the tests. I also experimented with different box sizes and particle counts for both tests to achieve a high resolution while keeping the total number of particles low.

Both test problems were then simulated with the SPH integrated into PKDGRAV3. The plots show that the results of the simulation are very close to the analytical solution for both tests. The densities, pressures and the speeds of the shocks match very well, any deviations are expected effects of SPH. The results of the simulations performed in this project prove that the SPH implementation in PKDGRAV3 is working correctly.

Over the course of this project, I ran into several problems and bugs, most of which were caused when the gravitational forces were disabled. The simulation had never been tested without gravity before. All of the problems have since been fixed, and there is now better documentation for PKDGRAV3.

The initial conditions created in this project are well suited for testing hydro-

dynamics codes, but they could be improved further. First, any remaining noise in the density and the pressure could be removed by iteratively relaxing the particles and then placing them back on the same isentrope. Second, the grid structure may not have fully disappeared, which means that the distribution is not entirely random and thus not a perfect glass.

The tests that were performed mainly focused on the propagation of shock waves. Further tests could be created using Rayleigh-Taylor or Kelvin-Helmholtz instabilities. It would also be interesting to run these tests with other hydrosolvers, for example MFM, to compare the results to those presented in this thesis.

Acknowledgments

There are many people without whom this project would not have been possible. I would like to thank

Prof. Robert Feldmann for inspiring me to do this work, for his supervision and guidance throughout the project, and for strengthening my interest in the field of computational astrophysics;

Prof. Siddhartha Mishra for being my supervisor from ETH Zurich;

Dr. Doug Potter for his patience when explaining the aspects of pkdgrav3, and for letting me come by whenever I had questions;

Thomas Meier for taking the time day and night to explain and fix any problems I ran into while running the simulations, for sharing his bachelor and master theses as references, and for his meticulous proofreading of my work;

Dr. Isaac Asensio for letting me use his python script for the analytical density of the Sedov blast wave;

Robin for carefully checking my writing, and Michelle for the emotional support.

Bibliography

[1] I. A. Asensio, C. D. Vecchia, D. Potter, and J. Stadel. Mesh-free hydrodynamics in pkdgrav3 for galaxy formation simulations. *Monthly Notices of the Royal Astronomical Society*, 519(1):300–317, nov 2022.

[2] K. M. Ferrière. The interstellar environment of our galaxy. *Reviews of Modern Physics*, 73(4):1031–1066, dec 2001.

[3] J. F. Hawley, L. L. Smarr, and J. R. Wilson. Numerical study of nonspherical black hole accretion. i. equations and test problems. *Astrophys. J.; (United States)*, 277:296–311, feb 1984.

[4] P. F. Hopkins. A general class of Lagrangian smoothed particle hydrodynamics methods and implications for fluid mixing problems. *Monthly Notices of the RAS*, 428(4):2840–2856, feb 2013.

[5] T. Meier. Equations of state for collision simulations. Bachelor's thesis, aug 2020.

[6] T. Meier. Extreme scale particle hydro methods. Master's thesis, dec 2021.

[7] J. J. Monaghan and J. C. Lattanzio. A refined particle method for astrophysical problems. *Astronomy and Astrophysics*, 149(1):135–143, August 1985.

[8] D. Potter, J. Stadel, and R. Teyssier. Pkdgrav3: Beyond trillion particle cosmological simulations for the next era of galaxy surveys, 2016.

[9] S. Roy. An introduction to fluid dynamics and numerical solution of shock tube problem by using roe solver, 2021.

[10] L. I. Sedov. Chapter iv - one-dimensional unsteady motion of a gas. In L. I. Sedov, editor, *Similarity and Dimensional Methods in Mechanics*, pages 146–304. Academic Press, 1959.

[11] G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31, 1978.

[12] J. G. Stadel. *Cosmological N-body simulations and their analysis*. PhD thesis, 2001.

[13] M. L. Timpe. *Emulation of Planetary-Scale Collisions*. PhD thesis, 2020.

[14] J. W. Wadsley, J. Stadel, and T. Quinn. Gasoline: a flexible, parallel implementation of treesph. *New Astronomy*, 9(2):137–158, 2004.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

A Test Suite for SPH Codes

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Riess	Armin

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 2.6.2023	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.