# SPH - FYSN33

Felix Filson

September 2025

## 1 Introduction

Smoothed particle hydrodynamics (SPH) is a numerical method developed to simulate complex astrophysical systems. As opposed to other numerical methods, SPH does not use grids. Instead, SPH utilises interpolation functions called kernels. The kernel represents the "sphere of influence" of each particle. In addition to simulating astrophysics, SPH can also be applied to the Navier-Stokes equations.

The purpose of this project was to apply SPH to two different situations. The first project aimed to solve the 1-D Sod's shock tube. This is a common test problem for computational hydrodynamics solvers. In the second project, the code was adjusted to work in 3-D and was used to simulate planetary collisions.

## 2 Theory

In the SPH method, the problem domain is constructed using particles without connectivity. This makes the method inherently mesh-free. Field functions, such as density or velocity, are expressed using integrals over the entire domain, weighted by a kernel-function. This kernel-function is bell-shaped and ensures that particles closer to the point at which the field-function is called contribute more than particles further away. The kernel function used in this project is

$$W(R,h) = \alpha_d \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3 & 0 \leq R < 1 \\ \frac{1}{6}(2-R)^3 & 1 \leq R < 2 \\ 0 & R \geq 2, \end{cases} \tag{1}$$

where $h$ is the smoothing length, $R$ is the absolute distance between two particles divided by the smoothing length $h$. In one dimension $\alpha_d = 1/h$ and in three dimensions, $\alpha_d = 3/(2\pi h^3)$.

### 2.1 1D Sod's Shock Tube Problem

The Sod's shock tube problem is a common test for computational fluid dynamics solvers. In this project, it served that purpose as well. The problem concerns a gas-filled tube. A membrane initially separates two gases of different densities and pressures. The membrane is then removed, and the system evolves into three distinct regions. They are a shock wave which moves into the less dense region, a rarefaction wave which moves into the higher density region, and finally, a contact discontinuity between the regions which moves into the low density region. In this project, the following set of differential equations was used to solve the problem

$$\begin{cases} \rho_i = \sum_{j=1}^{N} m_j W_{ij} \\ \frac{d\boldsymbol{v}_i}{dt} = -\sum_{j=1}^{N} \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{iju} \\ \frac{de_i}{dt} = \frac{1}{2} \sum_{j=1}^{N} \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) (\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \nabla_i W_{iju} \\ \frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{v}_i \end{cases}, \tag{2}$$

where $\rho$ denotes the density, $\boldsymbol{v}$ the velocity, $e$ the energy, and $x$ the position of the particle. $W_{ij}$ is the kernel function defined in equation 1. $\Pi_{ij}$ is the artificial viscosity introduced by Monaghan [1] and is needed to suppress unphysical oscillations as well as capture shock structures.

### 2.2 3D Planetary Collision

For 3D planetary collisions, self-gravitation between the particles has to be accounted for. In this project, the following formulas are obtained from a Ph.D. thesis by Peter Cossins [2]. When using a constant smoothing length $h$, the velocity term in the set of equations 2 is adjusted by adding

$$\left(\frac{d\boldsymbol{v}}{dt}\right)^{i}_{\text{Gravity}} = -\frac{G}{2}\sum_{j=1}^{N} m_j(\nabla_i\phi_{ij}(h_i) + \nabla_i\phi_{ij}(h_j))\frac{d\boldsymbol{x}}{r_{ij}}, \tag{3}$$

where $\phi_{ij}$ is the gravitational potential given by

$$\frac{d\phi(r,h)}{dr} = \begin{cases} \frac{1}{h^2}(\frac{4}{3}R - \frac{6}{5}R^3 + \frac{1}{2}R^4 & 0 \leq R < 1 \\ \frac{1}{h^2}(\frac{8}{3}R - 3R^2 + \frac{6}{5}R^3 - \frac{1}{6}R^4 - \frac{1}{15R^2} & 1 \leq R < 2 \\ \frac{1}{r^2} & R \geq 2. \end{cases} \tag{4}$$

## 3 Results

### 3.1 Sod's Shock Tube

The solver for Sod's shock tube was written in Python. The initial values as specified in the lab manual were then entered into an $N \times 6$ matrix. $N$ denoted the number of particles (400 in this case) and 6 represented the number of parameters tracked in this problem. On the left side of the membrane, 320 particles with density $\rho = 1$ were generated and on the right 80 particles with density $\rho = 0.25$. The derivatives were computed using helper functions. To make the code more efficient, the functions utilised broadcasting instead of loops. Thereafter, a fourth-order Runge-Kutta solver was built. This solver then evolved the system with a time step of 0.005 for a total of 40 time steps. The resulting plots are shown in figure 1. The figure shows four time steps of the evolution from $t = 0s$ until $t = 0.2s$. As expected, the density plot shows how the rarefaction moves to the more dense area towards the left and the shock wave moves into the less dense area. The initial contact discontinuity in the first plot also moves into the less dense area behind the shock wave, as expected.

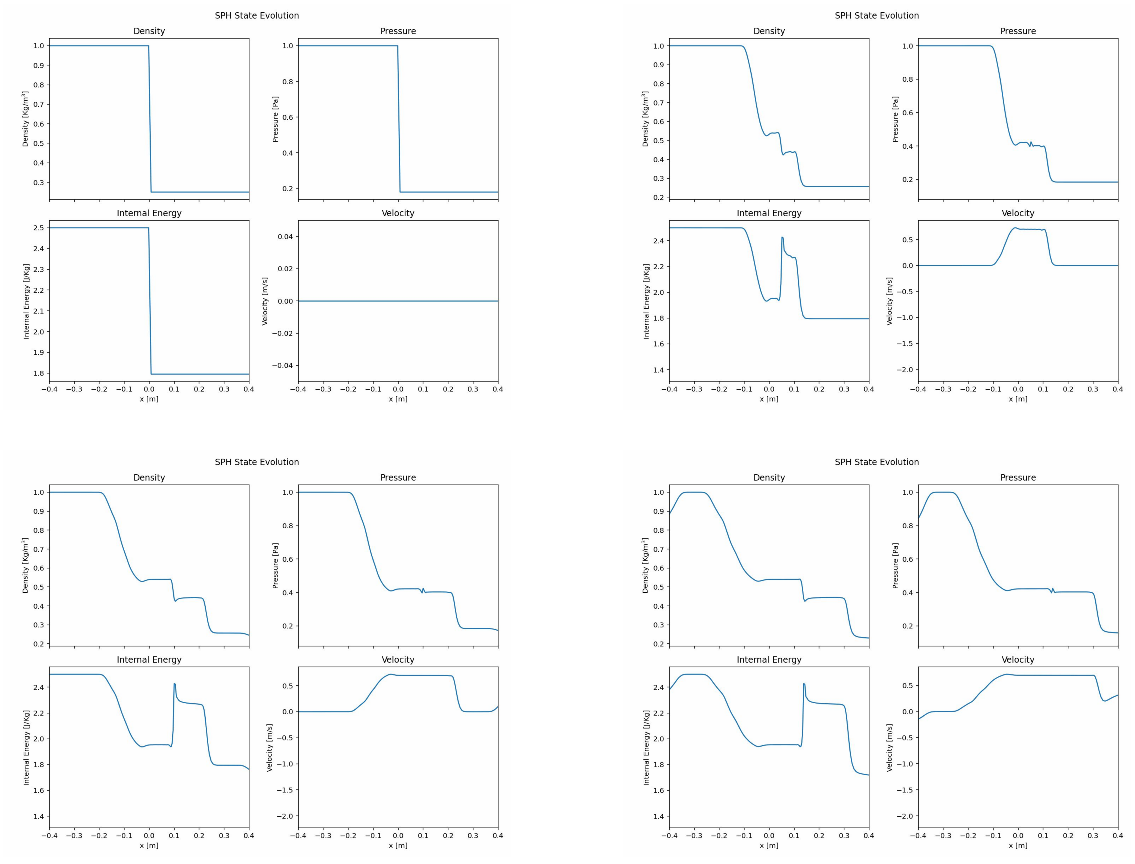

Figure 1: The images show the state evolution of density, pressure, internal energy and velocity at different time steps.

## 3.2 Planetary Collisions

Once the Shock Tube code was deemed to work, the code was adjusted to work in three dimensions. The first step involved using x,y,z components of position and velocity in the state vector. This resulted in an $N \times 10$ matrix. The helper functions were then also adjusted to work in 3D. Initially, the previously used RK4 function was used in the 3D case as well. However, this was not accurate enough and produced unstable planets. Therefore, Scipy's ivp solver was used instead. This module used an RK45 solver with adaptive time-steps. The state vector had to be flattened for use with the solver and thereafter unflattened for plotting. This fixed the problem and produced stable planets. Finally, rotations were added to planets and were then collided. As in the example provided in the lab manual. Two Jupiter-like planets with 600 particles each were collided, resulting in the images in figure 2. The result seemed to be in accordance with the images from the report meaning the simulations worked as expected.

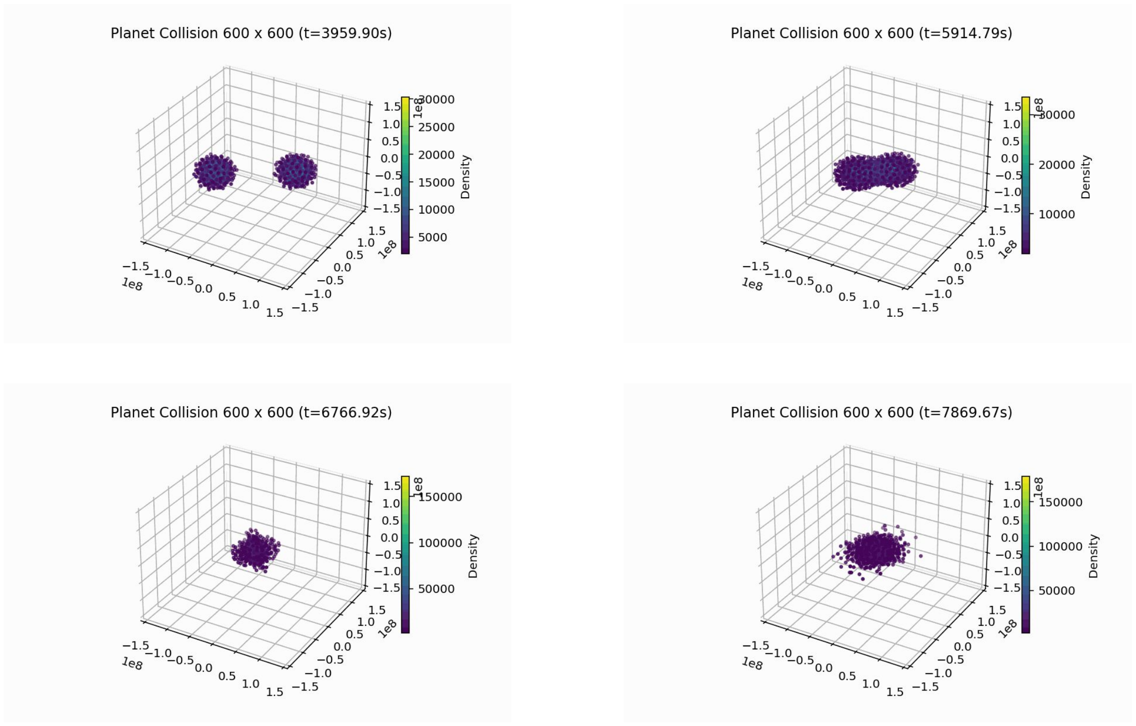

Figure 2: The images show the collision between two rotating Jupiter-like planets of 600 particles at different time steps.

## 4 Discussion

The two projects demonstrated the power and versatility of the SPH method. For the shock tube, SPH replicated the results in the lab manual reasonably well. Most importantly, the plots showed that the method had correctly reproduced a shock wave, a rarefaction wave, and a contact discontinuity. This showed that the kernel and overall method were working and could be applied to the 3D case. Due to the small scale and simple nature of this problem, running 40 time steps went very quickly. The simulation of the collision of the two planets with 600 particles also lined up with the result presented in the lab manual. The two planets have a glancing collision and then form a new, larger planet with some debris being permanently ejected. Therefore, the method once again seemed to work as expected. From a computational perspective, on the other hand, the planetary collision simulator was highly inefficient. The reason for this is due to the inverse-square term present in the gravitational potential $\phi_{ij}$ resulting in the problem being of $\mathcal{O}(N^2)$.

Another important point is the choice of integrator. My initial attempt with a fixed-step RK4 solver produced unstable results in the planetary collision. This happened because close encounters between particles required much smaller steps than the fixed time step could provide, which caused errors to accumulate. Switching to SciPy's RK45 solver, which uses adaptive time steps, fixed this problem. The solver automatically adjusted the time step depending on the strength of the interactions, which made the planets stable and the simulation physically consistent.

There were also several approximations made that limited the accuracy of the results. The brute-force calculation of gravity is computationally heavy and becomes impractical as the number of particles increases. The resolution was also very low. Each planet had only around 600 particles, which is far too few to capture realistic structure, shocks, or debris patterns. These details would also have been captured better if a variable smoothing length had been used instead of a constant one. The choice of time step was another compromise. Fixed time steps were fine for the simple 1D test but not for the 3D case, where adaptive stepping was necessary. Finally, while the cubic spline kernel is a standard and robust choice, other kernels could potentially increase accuracy.

Overall, this project demonstrated the implementation of SPH from scratch, testing accuracy in 1D and applying it to realistic astrophysical problems in 3D.

## 5 Conclusions

This project showed the versatility of the SPH method. It pretty accurately recreated shock, rarefaction, and contact discontinuities when applied to the shock tube problem. It also accurately simulated planetary collisions. The main drawback of the planetary simulation was the efficiency. Clearly, computing the self gravitation by brute force is not ideal. The efficiency of the simulation could be greatly improved by parallelising the problem. This would mean the simulation could also be made more accurate by, for example, using more particles to increase the resolution. For very inhomogeneous systems, a variable smoothing length could also be implemented in a more efficient simulation, which would also improve the accuracy.