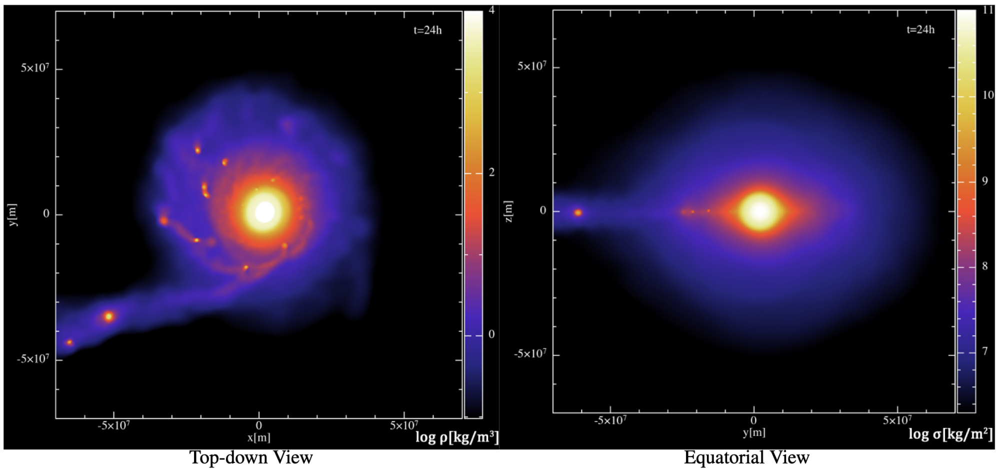
FYSN33 - Computational Physics

SPH Labs

Lab Manual



Top-down View   Equatorial View

Fig. 8. Density rendering of the system after 24 h of simulation time. A disk-like structure can be seen around an almost spherical inner planet. The color rendering on the *right-hand picture* represents the column density.

Figure credits: Lunar formation with SPH by R. Wissing and D. Hobbs

LUND UNIVERSITY

Lab Instructors:
TBD
tbd@fysik.lu.se

David Hobbs
david.hobbs@fysik.lu.se

1 Introduction to SPH

Smoothed particle hydrodynamics (SPH) was invented to simulate nonaxisymmetric phenomena in astrophysics (see Monaghan's review paper). They wanted a method that was easy to work with and could give reasonable accuracy. The SPH method satisfied these requirements. As a bonus they found that SPH was rugged, gave sensible answers in difficult situations, and could be extended to complicated physics without much trouble. The SPH method is a particle method and does not need a grid to calculate spatial derivatives. Instead, they are found by analytical differentiation of interpolation formulae. The SPH method can be applied to the Navier-Stokes equations and thus provides a powerful mechanism for solving fluid dynamics problems. The simple and intuitive form of the formula, coupled with detailed analysis, has allowed SPH to be extended to a wide variety of astrophysical problems.

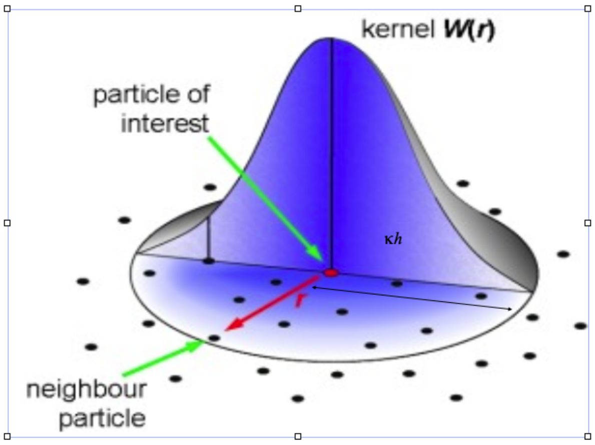


Figure 1: The SPH concept.

In the SPH method we use the following key ideas:

- The problem domain is represented by a set of arbitrarily distributed particles and no connectivity of the particles is needed (i.e. meshfree).
- An integral representation for the field functions is used and a kernel approximation in introduced. This provides stability and has a smoothing effect.
- The kernel approximation is then further approximated using particles. This is known as the particle approximation in SPH.
  - Done by replacing the integral representation of the function and its derivatives with summations over all corresponding values at the neighbouring particles in a local domain called the support domain (compact support).
  - Results in banded sparse matrices which can be solved efficiently
- The particle approximation is performed at each time step, so particles depend on the current local distribution of particles (adaptive). Therefore, SPH naturally handles problems with large deformation.
- The support domain must be sufficiently large and particles can be assigned mass and become physical material particles.

- The particle approximations are performed on all field functions to produce a set of ODE's in discretized form with respect to time only (Lagrangian). Such equations are conceptually simpler than the Eulerian equivalent.
- The ODE's are solved using an integration algorithm to achieve fast time stepping, and to obtain the time history of all the field variables for all the particles (dynamic).

2 Projects

This lab is split into three different projects, where all of them have to be addressed in order to pass. Using the theory learned in the lectures, you will implement the necessary equations and create animations for the different cases. The projects increase in difficulty from the first to last but are all doable within the course. The two projects are:

- Project 1: Solving the 1-D Sod's shock tube problem with SPH;
- Project 2: Solving the 3-D planetary collision problem with SPH.

For each of these projects you have to submit a short report which will include:

- Title page;
- Introduction section;
- Theory section;
- Results section;
- Discussion section;
- Conclusion section;
- Appendix section.

For each report you should include clearly labelled plots (and tables if necessary) of your results and normally also submit separate videos of the simulations (mp4 or gif). Project 1 corresponds to 60% and Project 2 corresponds to 40% of the marks. Project 1 but Project 2 can be merged into a single report, to save time and work, and then resubmitted with the extra sections.

Algorithm 1 Tip for the code: Use broadcasting to avoid loops and make your code efficient

```
procedure BROADCASTING
    # import numpy
    import numpy as np
    # Create some data
    x = np.linspace(7, 42, 100)
    y = np.linspace(96, 0, 100)
    # Use broadcasting to add every value of x to every value of y
    # (https://numpy.org/doc/stable/user/basics.broadcasting.htm)
    sumxy = x + y[:, np.newaxis]
    # If self-addition is not allowed, replace the diagonal elements
    # with zeros or whatever you need
    np.fill_diagonal(sumxy, 0)
```

2.1 Project 1: Solving the 1-D Sod's shock tube problem with SPH

The Sod shock tube problem (see wikipedia), named after Gary A. Sod, is a common test for the accuracy of computational fluid codes, like Riemann solvers which is a specific initial value problem composed of a conservation equation together with piecewise constant initial data which has a single discontinuity in the domain of interest. The problem was heavily investigated by Sod in 1978. The test consists of a one-dimensional Riemann problem with the following parameters, for left and right states of an ideal gas.

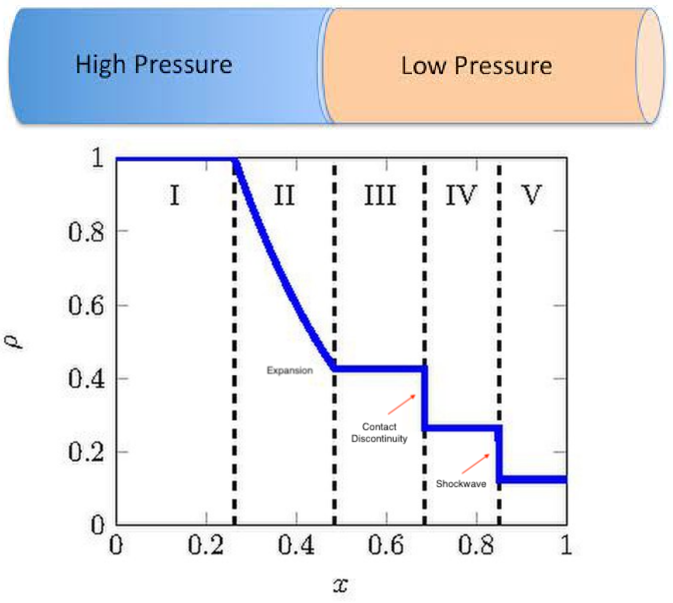


Figure 2: Density profile of Sod's shock tube problem at t=0.2 with the adiabatic γ=1.4, from wikipedia.

The Sod shock tube problem consists of a long straight tube filled with gas, which is separated by a membrane in two parts with different densities and pressures but are individually in thermodynamic equilibrium. When the membrane is taken away the following are produced:

- a shock wave - moves into the region of lower density;
- a rarefaction wave (reduction in density) - moves into the region of high density;
- a contact discontinuity - forms in the centre and travels into the low density region behind the shock.

In this first example, we use:

$$\begin{pmatrix} \rho_L \\ P_L \\ v_L \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.0 \\ 0.0 \end{pmatrix},$$
$$\begin{pmatrix} \rho_R \\ P_R \\ v_R \end{pmatrix} = \begin{pmatrix} 0.125 \\ 0.1 \\ 0.0 \end{pmatrix}. \tag{1}$$

where ρ is the density, P is the pressure, v is the velocity. The adiabatic $\gamma = \frac{C_P}{C_V}$ is the ratio of the heat capacity at constant pressure (C_P) to heat capacity at constant volume (C_V).

The time evolution of this problem can be described by solving the Euler equations for viscus flow, which leads to three characteristics, describing the propagation speed of the various regions of the system. Namely the rarefaction wave, the contact discontinuity and the shock discontinuity. If this is solved numerically, one can test against the analytical solution, and get information how well a code captures and resolves shocks and contact discontinuities and reproduce the correct density profile of the rarefaction wave.

For SPH we will use the following equations (other combinations are also possible):

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

If you initially set $\Pi_{ij} = 0$ you will see that un-physical oscillations develop. In a second test set up the artificial viscosity of Monaghan ($\alpha_\Pi = 1$, $\beta_\Pi = 1$, and $\varphi = 0.1h_{ij}$ to avoid divergence) as follows:

$$\Pi_{ij} = \begin{cases} \frac{-\alpha_\Pi \bar{c}_{ij}\phi_{ij} + \beta_\Pi \phi_{ij}^2}{\bar{\rho}_{ij}}, & \text{if } \mathbf{v}_{ij}\cdot\mathbf{x}_{ij} < 0 \\ 0, & \text{if } \mathbf{v}_{ij}\cdot\mathbf{x}_{ij} \geq 0 \end{cases} . \qquad (3)$$

$$\phi_{ij} = \frac{h_{ij}\mathbf{v}_{ij}\cdot\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2 + \varphi^2}, \qquad (4)$$

and the average density is

$$\bar{\rho}_{ij} = \frac{1}{2}(\rho_i + \rho_j) \qquad (5)$$

and the average speed of sound is

$$\bar{c}_{ij} = \frac{1}{2}(c_i + c_j) \qquad (6)$$

and the average smoothing length is

$$h_{ij} = \frac{1}{2}(h_i + h_j) \qquad (7)$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

For the project use the initial conditions given in table 1. The spacing of the particles can be calculated from their initial separation and the units are in SI. We use the equation of state for the ideal gas

$$p = (\gamma - 1)\rho e \qquad (8)$$

Table 1: Initial conditions for Sod's problem in the range $-0.6 \leq x \geq 0.6$.

	ρ	$\mathbf{v}$	e	p	δx	m	No. of Particles
$x \leq 0$	1	0	2.5	1	0.001875	0.001875	320
$x > 0$	0.25	0	1.795	0.1795	0.0075	0.001875	80

and the speed of sound is

$$c = \sqrt{(\gamma - 1)e} \tag{9}$$

$\gamma = \frac{c_P}{c_V} = 1.4$ is the ratio of specific heat (capacities). c_p and c_v are per unit mass and constant pressure and volume respectively. Set the time step to 0.005 and run the simulation for 40 time steps. It is generally better for this project to use a constant smoothing length due to the small number of particles used. An estimate can be made from the Eqn. 10 or it can be used as a way to calculate a variable smoothing length. η is a small fudge factor usually between 1.2 and 1.5 and d is the number of dimensions.

$$h = \eta \left(\frac{m_i}{\rho_i} \right)^{\frac{1}{d}} \tag{10}$$

No special treatment is used for the boundary as the shock wave has not yet reached it. Use a constant smoothing length which works better for small projects even though it is a poorer approximation. Use the cubic spline smoothing function (kernel) and its derivatives given in the lecture 2 notes. How you find nearest neighbours is an implementation issue and can be done very differently, talk to the lecturers in the course for help on this. If you implement all these equations and then run the project you should get something like the following for your results.

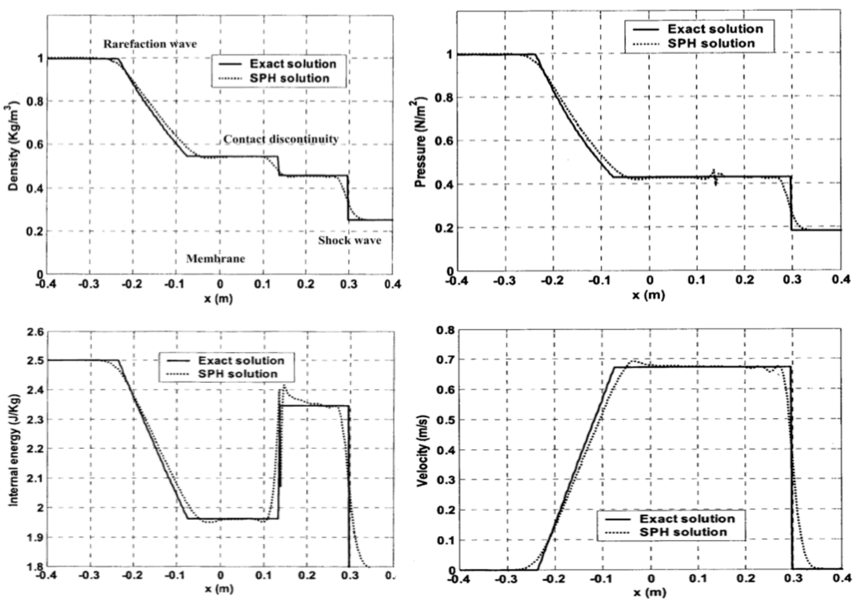


Figure 3: Results of Sod's shock tube problem after some time steps.

2.2 Project 3: Solving the 3-D planetary collision problem with SPH

Project 3 is essentially applying the same equations to a different problem. Sounds easy but it is challenging. Firstly you have to convert your 1-D code to 3-D, basically this means that position and velocity become vectors. It may be that you have programmed it this way from the start but that may not be the case and it needs testing. Secondly, you have to include self gravitation between the particles within the SPH formalism.

Use the following equations from the thesis: (P. Cossins, Chapter 3, Smoothed Particle Hydrodynamics, Ph.D. Thesis, Leicester 2010, handed out in class) (Note notation, he uses x we use R). For constant smoothing length the following simplified formula is valid and simplifies our calculations.

$$\frac{d\phi(r,h)}{dr} = \begin{cases} \frac{1}{h^2}\left(\frac{4}{3}R - \frac{6}{5}R^3 + \frac{1}{2}R^4\right), & \text{if } 0 \le R < 1 \\ \frac{1}{h^2}\left(\frac{8}{3}R - 3R^2 + \frac{6}{5}R^3 - \frac{1}{6}R^4 - \frac{1}{15R^2}\right), & \text{if } 1 \le R < 2 \\ \frac{1}{r^2} & \text{if } R \ge 2 \end{cases} . \tag{11}$$

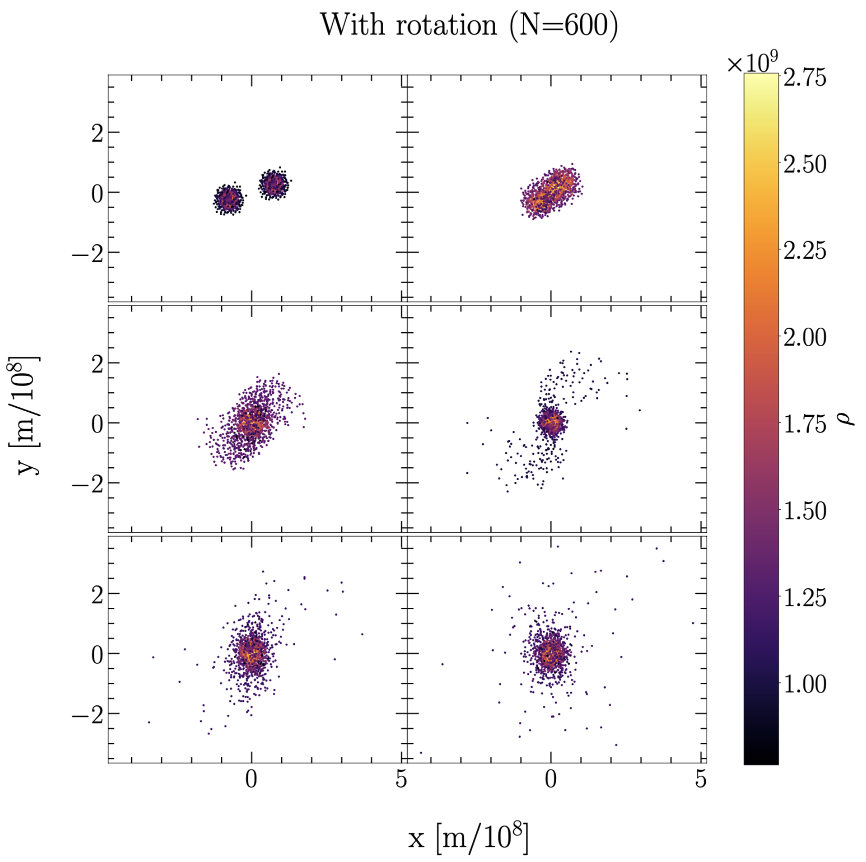


Figure 4: Example of two Jupiter like planets having a glancing collision and a spin. Credit: Nerea Gurrutxaga, Computational Astrophysics, 2022.

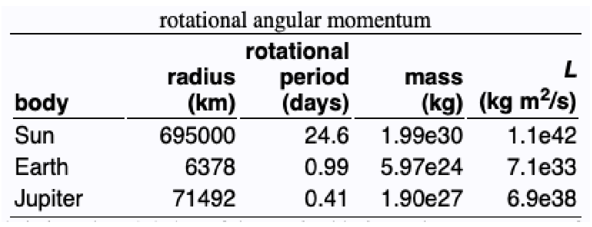
The third term in this equation is the inverse square law and applies to all particles thus making the problem of O(N^2). It greatly increases the computational load and slows the project down. There are nice efficient methods to handle this and they are discussed in lecture 6 but for your current projects it is probably best to use brute computation to solve the problem as is and python helps here provided the problem is not too large. Finally, we can then compute the gravitational force term to be added to the momentum equation as

$$\left(\frac{d\mathbf{v}}{dt}\right)^i_{\text{Gravity}} = -\frac{G}{2}\sum_j^N m_j(\nabla_i\phi_{ij}(h_i) + \nabla_i\phi_{ij}(h_j))\frac{d\mathbf{x}}{r_{ij}}\ . \tag{12}$$

where $R_{ij} = \frac{r_{ij}}{h} = \frac{|\mathbf{x}_i - \mathbf{x}_j|}{h} = \frac{|d\mathbf{x}|}{h}$.

The initial conditions for the project can be down loaded from canvas from the files PlanetXXXX.dat, where XXXX gives the number of particles in the single planet file. The data are in SI units and the format of the files is x, y, z, v_x, v_y, v_z, m, ρ, p. To do the simulation you first have to load a smaller planet to test your code. When you do so the planet will not be stable but should become stable after integrating with small time steps. It will oscillate and will expand slightly but should then settle down. Once you have a stable planet make a copy of it at another location and then send both planets flying towards each other with some velocity (which?) and some reasonable trajectory. It is also possible to add a spin to the planets. Making a planet spin can be done using the equation $\mathbf{v} = \omega \times \mathbf{r}$. For example you can give each planet an angular velocity of $\omega = (0, 0, \omega_z)$ where $\omega_z = \frac{2\pi}{T}$ and T is the period of rotation of the planet. Rotational angular momentum (not needed) is given by

$$L = (4\pi m r^2)/(5T) \tag{13}$$



rotational angular momentum

body	radius (km)	rotational period (days)	mass (kg)	*L* (kg m^2/s)
Sun	695000	24.6	1.99e30	1.1e42
Earth	6378	0.99	5.97e24	7.1e33
Jupiter	71492	0.41	1.90e27	6.9e38

Figure 5: Rotational angular momentum.

When all is done you should be able to make planetary collisions similar to the example in figure 4 which was done on a laptop. Access to larger computers would allow more realistic simulations. There are also more advanced codes out there but what you are implementing is sufficient for the basic science.

The project reports

Please typeset your project report as a PDF document, and upload it to canvas. The deadline for handing in the project report is the following Monday after labs finish.

The reports are typically around 5–10 pages (including plots) and video files can also be uploaded separately. Students can work together in labs must independently produce their own reports. For project 2 it is not necessary to write a new report. The student may update the Project 1 report to reduce the workload as much of the theory is common. This means that a new Theory subsection on how to add gravitation to SPH is needed and new Results, Discussion and Conclusion subsections are also needed. It is simpler if this new parts are placed in separate subsections to facilitate easier correction.

Please follow the general guideline of the report:

- **Title** contains the title and author information
- **Introduction** should contain a short general introduction to the subject and the purpose of the project.
- **Theory** should contain the basic theory underpinning the simulations.
- **Results** should contain your main plots which are clearly labelled. The section should explain how the results were obtained and their basic features.
- **Discussion** should contain a discussion of the results you obtained and their significance. You should discuss the approximations made while doing these numerical experiments.
- **Conclusion** briefly summarise the whole project and its main outcomes.
- **Appendix** to show any extra plots that were generated.

The project report should be written in your own words. Do not copy your answers from any other sources (like your classmate or the internet).