Computational Astrophysics Exercises — Assignments Set 10

Smoothed Particle Hydrodynamics – Toy Star

In this exercise, you will play around with a polytropic, 3D star model. We choose a sphere with radius $R = 0.75$ and mass content $M = 2$. We place particles with mass M/N randomly within this sphere and integrate the hydro equations using the SPH algorithm under consideration of a damping force and a simplified gravitational term. You can use this script* to generate the initial SPH particle distribution.
The structure of your code is as follows (please use the parameters as specified in table 1)

1. Read in input file.
2. while loop over all time steps, each substep includes the following steps
   a) Determine density for each particle (involves loop over all particles for each particle) using eq. (1).
   b) Determine sound speed for each particle.
   c) Determine pressure for each particle.
   d) Calculate acceleration for each particle due to the pressure force using eq. (2).
   e) Calculate linear acceleration force $\mathbf{a} = -\lambda \mathbf{x}$ with $\lambda = 2.01203286081606$† and $\mathbf{x} = (x, y, z)^T$.
   f) Calculate damping force $\mathbf{a} = -\nu \boldsymbol{v}$.
   g) Integrate velocities and positions of all particles using a simple Euler integrator (or better some 2nd order integrator).
   [f) for debugging purposes: plot particles/write additional output files]
3. Write final output file.

Since our algorithm involves loops over loops over all particles, it is computationally very expensive and gets real slow for moderate numbers of particles already. I highly recommend to use numba if you code in python‡.
For the SPH part of your code, use the following equation to calculate the density for each particle at first

$$\rho_i = \sum_{j=1}^{N_p} m_j W_{ij}, \tag{1}$$

where N_p denotes the number of particles within the smoothing length of particle i. With the density of all particles you can calculate the pressure gradient and its resulting acceleration for each particle using

$$\frac{d\boldsymbol{v}_i}{dt} = -\sum_{j=1}^{N_p} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}, \tag{2}$$

where the pressure is given by $p_i = K\rho_i^\gamma$.
What will happen during the simulation:

1. Since the initial particle distribution is not in hydrostatic equilibrium, the whole particle distribution will expand.

*`https://www.tat.physik.uni-tuebingen.de/~schaefer/f/mk_ini.py`
†Check `mk_ini.py` for the analytical λ.
‡on a Macbook Air (i5, 1.6 GHz), the python only version runs about 30 minutes without numba for 1000 particles

polytropic index	$n = 1$
polytropic exponent	$\gamma = 1 + 1/n$
polytropic constant	$K = 10^{-1}$
damping coefficient	$\nu = 10^{-1}$
smoothing length	$h = 0.2$
number of particles	$N = 1000$
simulation time	$t_{\text{end}} = 20$
fixed time step size	$\Delta t = 1 \times 10^{-2}$
kernel function	cubic B-spline

Table 1: Parameter settings for the Toy Star model simulation.

2. The particles will execute a damped oscillation around the final equilibrium state due to the attracting gravitational term $\lambda \boldsymbol{x}$ and the linear damping term $\nu \boldsymbol{v}$.

3. And finally, you should get a nice toy star. Plot the radial density distribution in the interval $[0, 2 \times R]$. The analytic solution is given by

$$\rho(r) = \left(\frac{\lambda}{2K(1+n)} \left(R^2 - r^2 \right) \right)^n ,$$

with $r = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$.

Please note that the production run with $N = 1000$ particles will run for quite a while. Depending on the CPU and the implementation, up to one hour in pure numpy/python.
If you like to know more, see `https://pmocz.github.io/manuscripts/pmocz_sph.pdf`.

in total (1/10 of all available points from compas summer term 2025)

Deadline:
Friday, 25 July at the time of the exercises class 👻 via email (including code) to `christoph.schaefer@uni-tuebingen.de`.