Doing a bit of Computational Astrophysics

BACKGROUND

Stars are made of high-temperature plasmas that can be understood as fluids under the influence of a gravitational field. In order to simulate them in the computer, we discretize them into fluid elements and we solve the Euler-Lagrange equations (and any additional physical processes required) on each one of those fluid elements. As an example, the file that you have just downloaded, contains 1,000,000 fluid elements -that we call *SPH-particles* or simply *particles*- that describe the center of a massive star.
If plotted, they look like this:

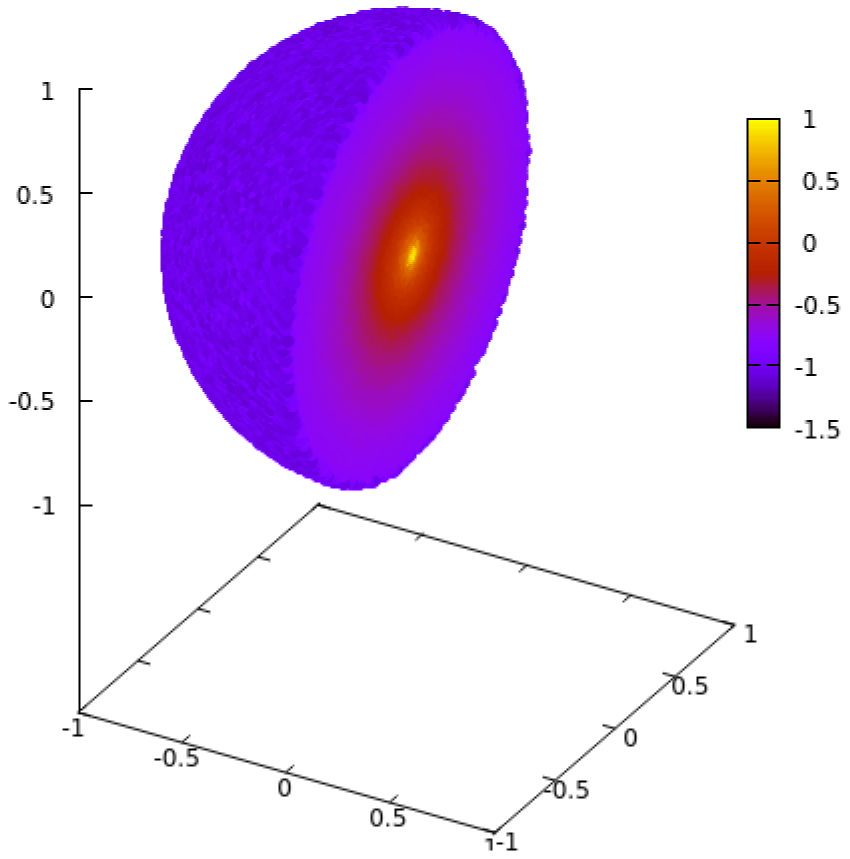

Figure 1: Longitudinal cut of a 3D massive star. Each point is a SPH-particle. Color represents density.

THE PROBLEM

In order, to solve the hydrodynamical equations we need to calculate physical properties as interpolations over close neighbors. Therefore, we have to find which particles are close to which ones. This is a non-trivial problem in three dimensions (3D) and there are elegant, fast solutions to do this. Nevertheless, we won't focus on how to find neighboring particles and assume that they have been found somewhere else. That's what the input file provides: the list of neighboring particles to each particle.

The code you have (main.cpp) calculates the density of each SPH particle doing the following summation:

$$\rho_i = \sum_{j=1}^{nv_i} m_j W_{ij}$$

where the index *i* runs for all the SPH particles and *j* runs for the list of neighbors of each *i* particle (nv_i). m_j is the mass of each neighboring particle (which in this particular case is the same for all particles) and W_{ij} is a weighting function similar to a Gaussian.

As you can see, the code's bottleneck is the double loop that runs for *i* and for *j* (nested loops). These should be the target of your optimizations.

Compilation: g++ -O2 -fopenmp main.cpp

Plot the results: gnuplot -p plot.gnu

Tips: In main.cpp, check that the file path for the input is correct. The program will output a file named density.txt which contains the results of the computations. You can add OpenMP directives to:

1) Enable parallel executions using threads
2) Enable GPU Offloading

You only need to modify one line (indicated with a comment) before the main loop.