1 Tracer particles

Tracer particles are to track the Lagrangian evolution of a model fluid using discrete particles. In hydrodynamical simulations based on an Eulerian grid (including CASTRO), thermodynamic variables at a given time are derived by solving the equations of motion of a fluid between cells. Therefore, in this scheme, the physical quantities that we can only access to are not discretized quantities at any given position, but rather average values over each cell. However, employing discrete particles, passively advected with the fluid flow, allows us to obtain local instantaneous thermodynamic variables, such as the temperature and the density, at well-defined positions, independent of the spatial resolution, i.e., the spatial cell size. This means that we can follow the evolution of the fluid at any given position and time.
CASTRO provides a tracer particle scheme with useful options in which, for example, the number and the initial positions of particles are flexibly determined according to the purpose of a given model.

2 Initializing the Particles

One must include the tracer particles in the `GNUmakefile` by setting

`USE_PARTICLES = TRUE`.

And the particles can be initialized via

castro.do_tracer_particles = 1

in the **inputs** file.

There are two ways to determine the initial positions of particles on a domain. You can either place the particles manually (Section 2.1) by providing an input file or randomly (Section 2.2) by setting some relavant variables.

2.1 manually-assigned initial positions

If one wants to investigate the evolution of fluid motions starting from specific positions (or a certain range of area or volume), one can manually specify the positions of particles by providing an input file containing the total number and the initial positions of particles. The input file should be in the same directory where your `inputs` file is located. The name of the input file is determined via :

particles.particle_init_file =*particle_file*

Here *particle_file* is the user-specified name of the file. The first line in this file is assumed to contain the number of particles. Each line after that contains the positions in a coordinate system adopted for your model. For 3-D cartesian coordinates,

x y z

For example, an input file for a model fluid with 6 particles in 2-D Cartesian coordinates may look like,

According to this input file, the 6 particles will be positioned at the same height (same y coordinate in the second column), equally spaced in x direction (the first column except for the particle number on the first line) from 3.2×10^8 cm to 1.4×10^9 cm.

2.2 randomly-assigned initial positions

Alternatively, one can randomly position particles. This may be used to characterize the global movement of a fluid. Even in this way of placing particles, however, the trajectory of each particle can be traced using the index individually assigned to each particle, which can be found in the output file (see Section 4).

To enable this option, set

castro.castro.particle_init_type = Random.

The number of particles are determined by setting:

castro.particle_initrandom_count = particle # (e.g., 10000).

3 Particle movement

There are currently two different ways in which particles can be moved, random motion and motion by self-gravity.
To enable the random motion, set

castro.particle_move_type = Random.

This updates the particle positions at the end of each coarse time step using a random number between 0 and 1 multiplied by $0.25\Delta x$ where Δx refers to the cell size .

On the other hand, by setting

castro.particle_move_type = Gravitational

the motion of particles are governed by their self-gravity.

4 Output file

The output files are stored in a directory whose name is determined by a variable `particles.timestamp_dir` . For example, if the variable is set as follows,

particles.timestamp_dir= *particle_dir*,

a directory *particle_dir* is automatically made with the directories for the main CASTRO output file (`plt****`) once a simulation starts and the particle output files are stored inside that directory.

The name of the output file consists of `Timestamp_` along with a number at the end. The number increases from 00 as more processors are involved in following the trajectories of particles. In parallel computating, a computational domain is divided according to the number of processors requested. Then each processor only follows the particles on the domain assigned to that processor and records their positions and velocities at any given time in a different output file. Since it is possible for particles to move from one domain to another during the evolution, its history can be stored in different files. More output files (with larger numbers at the end of the file name) can be produced as more processors follow the particles. Each particle is characterized by combining the integers in the first two columns in the outpux file, corresponding to a particle index assigned from each processor and the processor number. One way to identify the particles is consider the sum of the two integers as a global index for each particle.

By default, the output file constains the positions and velocities of all particles at a given time, meaning [3 + 2×dimensionality] columns. For example, for particles in a 3-D domain, the columns in the output file are,

index1 index2 x y z t v_{x} v_{y} v_{z} [ρ T]

The first two integers, as mentioned above, correspond to the particle index and the processor number. By setting the following,

particles.timestamp_temperature = 1,
particles.timestamp_density = 1,

one can also add the last two columns with the local temperature and local density at the particle positions, i.e. $[\rho \quad T]$. For example, let's consider 10 particles on a domain. If 4 out 10 particles are initially on a processor and the rest are on another processor, so two processors are tracking the particles and two output files are produced. In the output file written by the process with 4 particles, one can find that four lows are stored at the same time and each row corresponds to each particle info while in the other output file for the other 6 particles, 6 rows are stored at the same time.

If **particles.write_in_plotfile=1**, the particle data are stored in a binary file along with the main CASTRO output plotfile in directories `plt*****/Tracer/`.

The output profile can be post-processed by users. We shows plots below made from the output file as an example.

4.1 Run-time Screen Output

The verbosity written to the screen at run-time is constrolled by setting:
particles.v = 0 or 1 (default: 0)

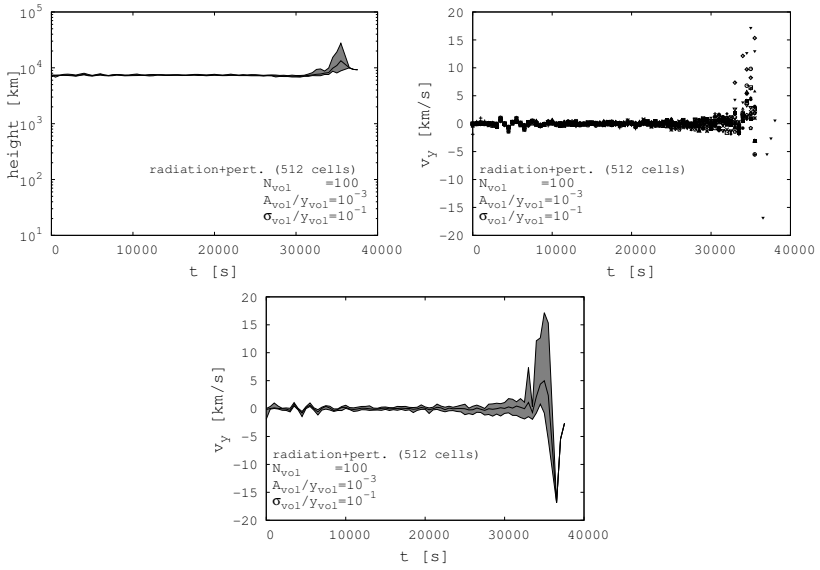

Figure 1: These plots show an example of how the output data can be visualized for particles advected with a fluid motion in a model planetary atmosphere. *Left* panel : the time evolution of the height of particles. The solid line represents the average height and the shaded regions indicate the maximum and minimum values of the height at a given time. *Right* panel : the velocities in y-direction ($v_{\rm y}$) of all particles. Different dot types means different particles. *Bottom* panel : same as the *right* panel. However, this shows the average velocity (solid line) and the maximum and minimum velocities at a given time (shaded region). The *left* and *bottom* panels are made after post-processing the data.