ASTR 206 Lecture Notes

Simeon Bird

November 27, 2023

1 Gas and Hydro Codes

1.1 Fluid Mechanics

Fluid mechanics in astrophysics deals with the study of inviscid fluids where self-gravity is important. A classic example of an inviscid fluid is slow moving water in a pipe. One complexity in astrophysics is that fluids are often supersonic.
I will use index notation with implicit summation throughout the below.

1.1.1 Microphysics

A fluid is a set of molecules colliding frequently. Their true phase space is described by the Boltzmann equation, but it can be approximated by the evolution of averaged 'bulk' properties like fluid velocity u_i, density ρ and pressure P (strictly a tensor P_{ij} but we will assume isotropic pressure).
An important thing to know is that for fluid mechanics the element speed is not the sound speed and not the equation of state. Thus:

$$|u|^2 \neq c_s^2 = \frac{\partial P}{\partial \rho} \neq \frac{P}{\rho} \tag{1}$$

For most fluids, $c_s^2 > |u|^2$.

1.1.2 Material Time Derivative

We also want to know how quantities change for objects comoving with the fluid. For this we need the material time derivative, a differential operator denoted by D.

$$\frac{DF}{Dt} = \frac{\partial F}{\partial t} + u_i \nabla_i F \tag{2}$$

The first term is the derivative with respect to local time, and the second term is the change in F due to the fluid motion. To see why this is a total derivative, we can write it out as:

$$\frac{DF}{Dt} = \frac{\partial F}{\partial t} + \frac{\partial x_i}{\partial t}\frac{\partial}{\partial x_i}F \tag{3}$$

These two derivative operators, Dt and ∂t, lead to two numerical pictures of fluid mechanics: the Eulerian picture and the Lagrangian picture. The Eulerian picture discretizes the fluid at a constant location in space, so that the time derivatives are ∂t. The Lagrangian picture discretizes a fluid **moving with the averaged bulk flow**, so that the time derivatives are Dt.
Mathematically, either picture is a good choice, but each picture has numerical advantages and disadvantages. A lot of things are conserved when evolved in the Lagrangian picture. In particular, $\frac{D\rho}{Dt} = 0$ for incompressible flows.
Numerically it definitely matters! The Eulerian picture is easier to implement, but the Lagrangian picture has advantages when self-gravity is important.

1.1.3 Mass Conservation

We need to ensure mass conservation in the fluid. We do this be enforcing that the change in mass inside an infinitesimal volume is equal to the change in density across the boundary of the volume.

$$\frac{\partial \rho}{\partial t} + \nabla_i(\rho u_i) = 0 \tag{4}$$

or using the Lagrangian picture:

$$\frac{D\rho}{Dt} + \rho \nabla_i u_i = 0 \tag{5}$$

This is the mass equation. A similar but longer argument leads to the momentum equation:

$$\rho \frac{Du_i}{Dt} = -\nabla_j P_{ij} + g \tag{6}$$

which is the inviscid Euler equation (g is an external gravitational field). We have neglected viscosity in this equation!

1.2 Hydro Simulations

The problem is to discretize the above continuous PDEs into discrete chunks in a way which minimises the truncation error. In practice, since we do not know the truncation error before doing the computation, we instead check that increasing the number of resolution elements gives roughly the same answer. Notice that pressure forces are local, which makes this fundamentally different from a gravity code: we only need to do interactions between nearby regions.

1.2.1 Eulerian "Fixed Grid" Hydro

Divide space into a stationary grid. Usually a cubic N-D grid is used. Examples include Nyx and Athena.

1. Assume each cell has constant internal density.

2. Work out mass flux across surfaces.
3. Work out pressure flux at each surface.
4. Update and repeat.

Advantages:

1. Easy to implement
2. Well defined limit as $N \to \infty$ which is easy to test.
3. Obvious choice to model, eg, atmospheric models or waver waves.
4. Scalable: can be set up so neighbours are just one array element further and operations are local.

Disadvantages:

1. Works poorly when the density varies spatially by more than an order of magnitude. Spatial resolution is limited to the size of a grid cell. This makes it totally unsuitable for galaxy formation.
2. Inaccurate for supersonic flows.

Two possible solutions:

1. A fixed but cleverly shaped grid which puts more cells in higher density regions (if you know where those are in advance). Modelling the flow over an aircraft wing is a useful application.
2. Adaptive Mesh Refinement.

1.2.2 Adaptive Mesh Refinement

The idea of adaptive mesh refinement is that when collapsing gas reaches a certain density, the fixed grid cell is replaced with a finer sub-grid. If the density increases further, a yet finer grid is placed. The grid is still stationary, but it has higher resolution. Refined grids are never removed.
Advantages:

1. Improves resolution in high density areas.
2. Classic application is simulations of single galaxies.

Disadvantages:

1. Ideally need to place mesh in advance of something happening, which can be complicated.

2. Memory usage is unbounded and uncontrolled. Scalability properties are bad.

3. When you have a lot of galaxies and they are moving around, like in a cosmological volume, you need to have a lot of refined grids and it becomes complicated.

4. Outflows are difficult to model. In an outflow, dense gas will move from the fine region to the less dense region, and the fluid properties will be smoothed out. This does not really fit well with an adaptive mesh.

5. Momentum conservation is not explicit going from high to low refinement.

Examples: Enzo, Nyx, Athena.
See `https://levelup.gitconnected.com/create-your-own-finite-volume-fluid-simulation-with`
for a simple implementation in python.

2 Smoothed Particle Hydrodynamics

(Monaghan 1992, Springel 2005)
We would like a technique that fits well with the smooth density hierarchy of galaxies, and ideally fits into the particle/tree structure of the gravity code. That technique is Smoothed Particle Hydrodynamics (SPH), which is implemented into MP-Gadget.
In SPH we discretise the fluid in cells of constant mass. Essentially we model the elements as small balls, which interact with each other. We achieve higher effective resolution in dense areas because there is more mass, and thus more balls.
These balls are not point particles, but smooth blobs with a defined size. We define an internal density given by a smoothing kernel $W(|r - r'|, h)$, where h is the **SPH smoothing length**. For a fluid quantity, for example ρ, define:

$$\rho(r) = \int_{R^3} \rho(r')W(r - r', h)dr' \tag{7}$$

In other words, we are using a Greens' function for the density, and identifying the function as the smoothing kernel of the particle. To be a valid Greens' function we need the kernel to have certain properties:

$$\int_{R^3} W(r - r', h)dr' = 1 \tag{8}$$

$$\lim_{h\to 0} W(r, h) = \delta(r) \tag{9}$$

ie, it should be normalised and a delta function in the small h limit.
When we discretise into particles, the integral becomes a summation: (using $\rho dV = dm$):

$$\rho(r) = \Sigma_{i\in \text{part}} m_i W(r - r_i, h_i) \tag{10}$$

Because m_i does not depend on position, we can take gradients by differentiating the kernel function W. This is a kernel trick!

$$\nabla\rho(r) = \Sigma_{i\in \text{part}} m_i \nabla W(r - r_i, h_i) \tag{11}$$

More generally for some scalar A:

$$A(r) = \Sigma_{i\in\text{part}} \frac{m_i A_i}{\rho_i} W(r - r_i, h_i) \tag{12}$$

It remains to pick the kernel function. A Gaussian is a good choice, because it is analytically tractable, and you can show the integral converges. However, it is numerically a poor choice, because it means that you have turned a local (pressure) force into a long-range force, forcing an integration over all particles and making the method $\mathrm{O}(N^2)$.
A numerically more efficient choice is a kernel with **compact support**. To ensure the kernel is continuous, we use a cubic spline:

$$W(r, h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6(r/h)^2 + 6(r/h)^3 \qquad 0 < r/h < 1/2 \\ 2(1 - (r/h))^3 \qquad 1/2 < r/h < 1 \\ 0 \qquad r/h > 1 \end{cases} \tag{13}$$

Since we now have compact support, we need to choose h. Note that because we have a kernel trick, the value of h is a numerical choice: the mathematics works irrespective of what h is. The integration is less noisy the more particles are included, at the cost of spatial resolution. To ensure that we smoothly increase resolution in dense areas, we choose h so that it encloses a roughly fixed number of particles, which we will call $N_{sph} \approx 100$.

$$\frac{4\pi}{3} h_i^3 \rho_i = N_{sph} \bar{m} \tag{14}$$

$$\rho(r) = \Sigma_{i\in\text{part}} m_i W(r - r_i, h_i) \tag{15}$$

Notice that this is an implicit equation! In practice we solve it iteratively using a Newton-Raphson method. First we find h, then ρ, then h again until convergence. Gadget makes it faster by estimating the change in h each timestep and using that as an initial guess.
Notice that the use of a compact kernel has introduced another limit whose convergence needs to be evaluated: $N_{\text{sph}} \to \infty$. Now we need an equation of motion, and thus a pressure gradient. Based on the foregoing, the natural choice might seem to be:

$$\rho_a \nabla P_a = \Sigma_b m_b (P_b - P_a) \nabla_a W_b(r_a - r_b, h_a)\,. \tag{16}$$

However this is a poor choice because it is not symmetric and thus does not obey Newton's third law and nor conserve momentum. The truncated kernel means that SPH density estimates are noisy, so that a lack of momentum conservation can become serious quickly.
Instead we can use various symmetrised forms of this equation. Different codes make different choices, but the one used in Gadget-2 is:

$$\frac{dv_i}{dt} = \Sigma_{i=1}^{N} m_j \left(f_i \frac{P_i}{\rho_i^2} \nabla_i W_{ij}(h_i) + (i \leftrightarrow j) \right) \tag{17}$$

$$\frac{1}{f_i} = 1 + \frac{1}{3} \frac{d \log \rho}{d \log h} \tag{18}$$

which explicitly conserves momentum (and entropy) in an adiabatic flow.
This in turn does badly for shocks, where entropy is not locally conserved. This is partially fixed with an extra artificial friction "viscosity" force, which approximates the force error in these circumstances.
An improved SPH technique is that from Hopkins 2013, which changes the kernel to a larger, quintic kernel, which better approximates a Gaussian, at the cost of some speed and resolution. In addition it uses a better gradient estimator which explicitly conserves energy and includes terms due to ∇h.

3 Unstructured Mesh Methods (AREPO)

(see Springel 2010)
It would be nice to create a technique which combines the best parts of both grid and SPH codes. The specific aspect of SPH codes we want is to smoothly increase the resolution in dense areas, whereas the specific aspect of grid codes we want is to use mass and momentum fluxes to model the hydro equations.
The elegant solution is to use an unstructured mesh. In Arepo this is generated by a Voronoi tessellation. In Gizmo it uses the same smoothed balls h as in SPH.
Each SPH particle now becomes a mesh generating point. All space that is closer to grid point i than any other grid point is part of the mesh cell around that point. Flux transfer between two grid cells is done along a vector joining the two grid points, in a frame comoving with the grid point (so it is Galilean invariant and somewhat Lagrangian). The motion of the grid cells models the overall fluid flow, the fluxes model small-scale diffusion.
Why was this not done before? A problem with completely Lagrangian codes is "mesh tangling". In these codes, a mesh cell was supposed to contain specific molecules, and there was no motion between cells. However, in turbulent areas this meant that the cells could become oddly stretched, with multiple neighbours. Neighbour finding thus became slow. In Arepo the mesh is not physical: the mesh cells can move, inducing a fluid flow across the mesh cell boundary even in the absence of particle velocity.
This allows the mesh to be kept simple. Arepo adds an extra artificial force to move the mesh cells relative to the flow, with the aim of keeping the mesh cells as circular as possible. Notice the resolution is now 1 grid cell again, rather than 100 SPH particles. Because the mesh moves, conservation is not guaranteed, but can be imposed by careful choice of flux. Shocks are fine and modelled!
One cool feature is the ability to refine cells further if they are cooling fast, called super-Lagrangian refinement.