ASTR 206 Lecture Notes

Simeon Bird

September 24, 2024

1 Gravitational Simulations

This section is about N-Body codes, or the problem of simulating gravitational interactions. An important difference from other types of physical simulations is that gravity is a non-local interaction, which significantly changes the types of simulation models that can be used. Note that there are two types of N-body code. One simulates structure formation, where an initially uniform density dark matter fluid in an expanding background is allowed to evolve under self-gravity. We are concerned with the other, which models the evolution of small clusters of stars or black holes.

The first N-body simulation was performed by Eric Holmberg in 1941. The forces were computed using O(N) effort by placing light bulbs in a room and computing accelerations using light fluxes onto a photon cell. The bulbs were then moved by hand.

In this problem:

1. Each star/BH is a point particle.
2. The force on each star is given by Newtonian gravity. For particle j the force from other particles $i \neq j$ is:

$$a_j = \Sigma_i \frac{GM_i}{(r_i - r_j)^2} \tag{1}$$

Why can we treat a star as a point particle? Because even in the densest cluster the distance between stars is usually much larger than the size of a star.

It is well known that a 2-body problem is solvable analytically, but the 3-body problem is not, except under very special conditions. It is furthermore chaotic: the final state depends sensitively on the initial positions and velocities. The N-body problem is also analytically intractable.

1.1 Forces

Let's first talk about fast ways to compute the forces. The obvious algorithm is something like:

```
for j in particles:
   acc_j = 0
   for i in particles:
     if i != j:
       acc_j += G M_i / (r_i - r_j)^2
```

but this is slow for large numbers of particles: for N particles this needs $N \times N$ operations and thus is said to be $O(N^2)$. Notice that this notation drops a constant prefactor: it could be $2N^2$, $3N^2$, or even $200N^2$ operations, and it does not count how fast the operation is. Nevertheless, when $N \sim 10^{12}$, the cost is large.

1.2 Timestepping

We also need a discrete timestepping rule to move the system from time t_i to time $t_{i+1} = t_i + \delta t$. This is a 6D system, 3 position variables (x_i) and 3 velocity variables (v_i), moving under Hamiltonian dynamics. A reasonable choice might seem to be:

$$v_{i+1} = v_i + a_i \delta t \tag{2}$$
$$x_{i+1} = x_i + v_i \delta t \tag{3}$$

but this is in fact a terrible choice!

The gravitational system follows energy-conserving Hamiltonian dynamics. Because it is not robust to non-Hamiltonian perturbations, the discrete timestepping rule must also follow Hamiltonian dynamics, and so each step should preserve energy. Why does this choice not preserve energy?

More precisely, the thing that needs to be conserved is phase space volume. We can define a Hamiltonian in terms of canonical coordinates $p = mv$ and x:

$$H = \frac{1}{2}\frac{p^2}{m} + \Phi(x) \tag{4}$$

where Φ is the gravitational potential. Then we have the dynamics:

$$\dot{x} = \frac{\partial H}{\partial p} = \frac{p}{m} \tag{5}$$
$$\dot{p} = -\frac{\partial H}{\partial x} = -\frac{\partial \Phi}{\partial x} \tag{6}$$
$$\tag{7}$$

Conserving phase space volume is written as $dp \times dx = 0$ and can be understood as "close trajectories stay close".

The Euler method from Eq. 3 can be written as a rule from (p,x) to (p', x'):

$$\begin{pmatrix} p' \\ x' \end{pmatrix} = \begin{pmatrix} 1 & \frac{\delta t}{x}\frac{\partial H}{\partial x} \\ \delta t & 1 \end{pmatrix} \cdot \begin{pmatrix} p \\ x \end{pmatrix} \tag{8}$$

and $(p' - p) \times (x' - x)$ is not zero.

The fix to this is to use a **symplectic integrator**, notably the **leapfrog**. The idea is to split it into two sections, one part which evolves dx and the other part which evolves dp. Since at any step either dx or dp is zero, the product is trivially zero.

Any Hamiltonian system ideally wants a symplectic integrator: the Hamiltonian Monte Carlo used in pymc also uses symplectic integration for steps.

We define two operators for the gravitational system, one which evolves the kinetic term (the "kick", K) and one which evolves the position term (the "drift", D). Each of these is individually phase space preserving, and we apply them in turn.

$$K(\delta t) : v_i \to v + a\delta t \tag{9}$$
$$D(\delta t) : x \to x + v\delta t \tag{10}$$
$$\tag{11}$$

The full timestep is kick-drift-kick, which splits the kick into two and is:

$$K(\delta t/2)D(\delta t)K(\delta t/2) \tag{12}$$
$$\tag{13}$$

Notice that when the drift is applied the velocity that is used is evaluated at $t + \delta t/2$. It would not be consistent to examine the energy of the system in the middle of the step, as the velocity and positions of the particles are at different times.

Choosing timestepping schemes which preserve energy is a good idea in computational astrophysics. The timestep itself can be set using multiples of the local dynamic time, so that particles with larger acceleration changes have smaller timesteps.

1.3 Binary Formation

A common feature of gravitational systems is the formation of binaries. One mechanism is three body interactions. Here one of the objects takes away the angular momentum of the system, leaving the other two bound.

If the binary is tight, the timestep needed to evolve it is small, and thus many more forces must be worked out. This is especially wasteful as the evolution of an isolated binary has an analytic solution. Some codes detect isolated binary systems and evolve them forward using the analytic solution. Many high performance codes allow the timesteps for each particle to be set individually, so that the binary does not slow down the whole simulation. An issue that arises is how to model interactions between particles on different timesteps. For this see the Gadget-4 paper `https://arxiv.org/abs/2010.03567`.

2 Structure Formation

For structure formation problems, things are similar, but instead of simulating point particles we are simulating an (almost) uniform density fluid. So the approximation that we made of point particles is no longer good. Why can we still use N-body techniques?

We can extensively use Birkhoff's theorem instead: the gravitation from any distribution of matter can be replaced by a point particle at the center of mass if the angular size is sufficiently small.

3 Write our own N-body code

Let's now write our own simple N-body code. You may write in python (or any other language that you prefer), and the code need use only the easiest algorithms.

- Write the code to compute forces. Use direct summation.
- Generate an initial distribution of particles. Give them a velocity, a position and a unique ID. Make a parameter for the number of particles.
- Implement the kick-drift-kick symplectic time integrator for time integration.
- Implement a criterion for the timestep length, scaling like the acceleration. Check how converged the results are with timestep size.
- Implement a test that checks the results against the evolution of a stellar binary.

Run this code! You can confirm that it scales as $O(N^2)$.

4 Some Literature

- Advanced N-body solver: `https://rebound.readthedocs.io/en/latest/`
- Springel 2005, Gadget-2: `https://arxiv.org/abs/astro-ph/0505010`
- Aarseth, Sverre "Gravitational N-body Simulations" `https://doi.org/10.1017/CBO9780511535246`
- Binney and Tremaine "Galactic Dynamics" `https://doi.org/10.2307/j.ctvc778ff`