4 Tree algorithms

Once we have discretized a collisionless fluid in terms of an N-body system, two questions come up:

1. How do we integrate the equations of motion in time?

2. How do we compute the right hand side of the equations of motion, i.e. the gravitational forces?

For the first point, we can simply use one of our ODE integration schemes, preferably a symplectic one since we are dealing with a Hamiltonian system. The second point seems also straightforward at first, as the accelerations (forces) can be readily calculated through *direct summation*:

$$\ddot{\mathbf{x}}_i = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{x}_i - \mathbf{x}_j)^2 + \epsilon^2]^{3/2}} (\mathbf{x}_i - \mathbf{x}_j). \tag{4.1}$$

This calculation is *exact*, but for each of the N equations we have to calculate a sum with N partial forces, yielding a computational cost of order $\mathcal{O}(N^2)$. This quickly becomes prohibitive for large N, and causes a conflict with our urgent need to have a large N!

Perhaps a simple example is in order to show how bad the N^2 scaling really is in practice. Suppose you can do $N = 10^6$ in a month of computer time, which is close to the maximum that one may want to do in practice. A particle number of $N = 10^{10}$ would then already take of order 10 million years.

We hence need faster, approximative force calculation schemes. We shall discuss three different possibilities:

- Hierarchical multipole methods ("tree-algorithms")

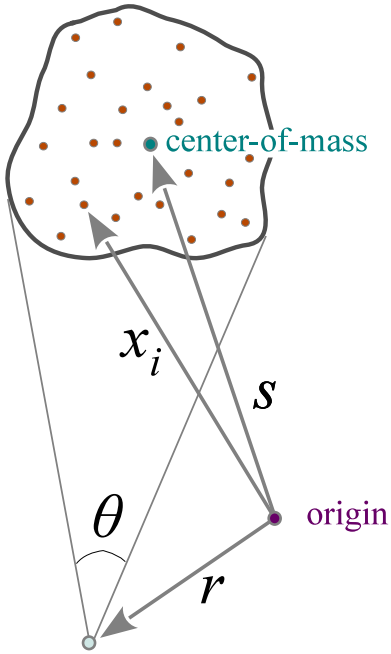
- Fourier-transform based methods ("particle-mesh algorithms")

- Iterative solvers for Poisson's equation ("relaxation methods", multigrid-methods)

Various combinations of these approaches may also be used, and sometimes they are also applied together with direct summation on small scales. The latter may also be accelerated with special-purpose hardware (e.g. the GRAPE board), or with graphics processing units (GPUs) that are used as fast number-crushers.

4.1 Multipole expansion

The central idea is here to use the multipole expansion of a distant group of particle to describe its gravity, instead of summing up the forces from all individual particles.



The potential of the group is given by

$$\Phi(\mathbf{r}) = -G \sum_i \frac{m_i}{|\mathbf{r} - \mathbf{x}_i|}, \tag{4.2}$$

which we can re-write as

$$\Phi(\mathbf{r}) = -G \sum_i \frac{m_i}{|\mathbf{r} - \mathbf{s} + \mathbf{s} - \mathbf{x}_i|}. \tag{4.3}$$

Now we expand the denominator assuming $|\mathbf{x}_i - \mathbf{s}| \ll |\mathbf{r} - \mathbf{s}|$, which will be the case provided the *opening angle* θ under which the group is seen is sufficiently small. We can then use the Taylor expansion

$$\frac{1}{|\mathbf{y} + \mathbf{s} - \mathbf{x}_i|} = \frac{1}{|\mathbf{y}|} - \frac{\mathbf{y} \cdot (\mathbf{s} - \mathbf{x}_i)}{|\mathbf{y}|^3} + \frac{1}{2} \frac{\mathbf{y}^T \left[3(\mathbf{s} - \mathbf{x}_i)(\mathbf{s} - \mathbf{x}_i)^T - (\mathbf{s} - \mathbf{x}_i)^2 \right] \mathbf{y}}{|\mathbf{y}|^5} + \ldots, \tag{4.4}$$

where we introduced $\mathbf{y} \equiv \mathbf{r} - \mathbf{s}$ as a short-cut. The first term on the right hand side gives rise to the monopole moment, the second to the dipole moment, and the third to the quadrupole moment. If desired, one can continue the expansion to ever higher order terms.

These multipole moments then become properties of the group of particles:

$$\text{monopole:} \quad M = \sum_i m_i \tag{4.5}$$

$$\text{quadrupole:} \quad Q_{ij} = \sum_k m_k \left[3(\mathbf{s}-\mathbf{x}_k)_i(\mathbf{s}-\mathbf{x}_k)_j - \delta_{ij}(\mathbf{s}-\mathbf{x}_k)^2\right] \tag{4.6}$$

The dipole vanishes, because we have done the expansion relative to the center-of-mass, defined as

$$\mathbf{s} = \frac{1}{M}\sum_i m_i \mathbf{x}_i. \tag{4.7}$$

If we restrict ourselves to terms of up to quadrupole order, we hence arrive at the expansion

$$\Phi(\mathbf{r}) = -G\left(\frac{M}{|\mathbf{y}|} + \frac{1}{2}\frac{\mathbf{y}^T\mathbf{Q}\mathbf{y}}{|\mathbf{y}|^5}\right); \quad \mathbf{y} = \mathbf{r} - \mathbf{s}, \tag{4.8}$$

from which also the force can be readily obtained through differentiation. Recall that we expect the expansion to be accurate if

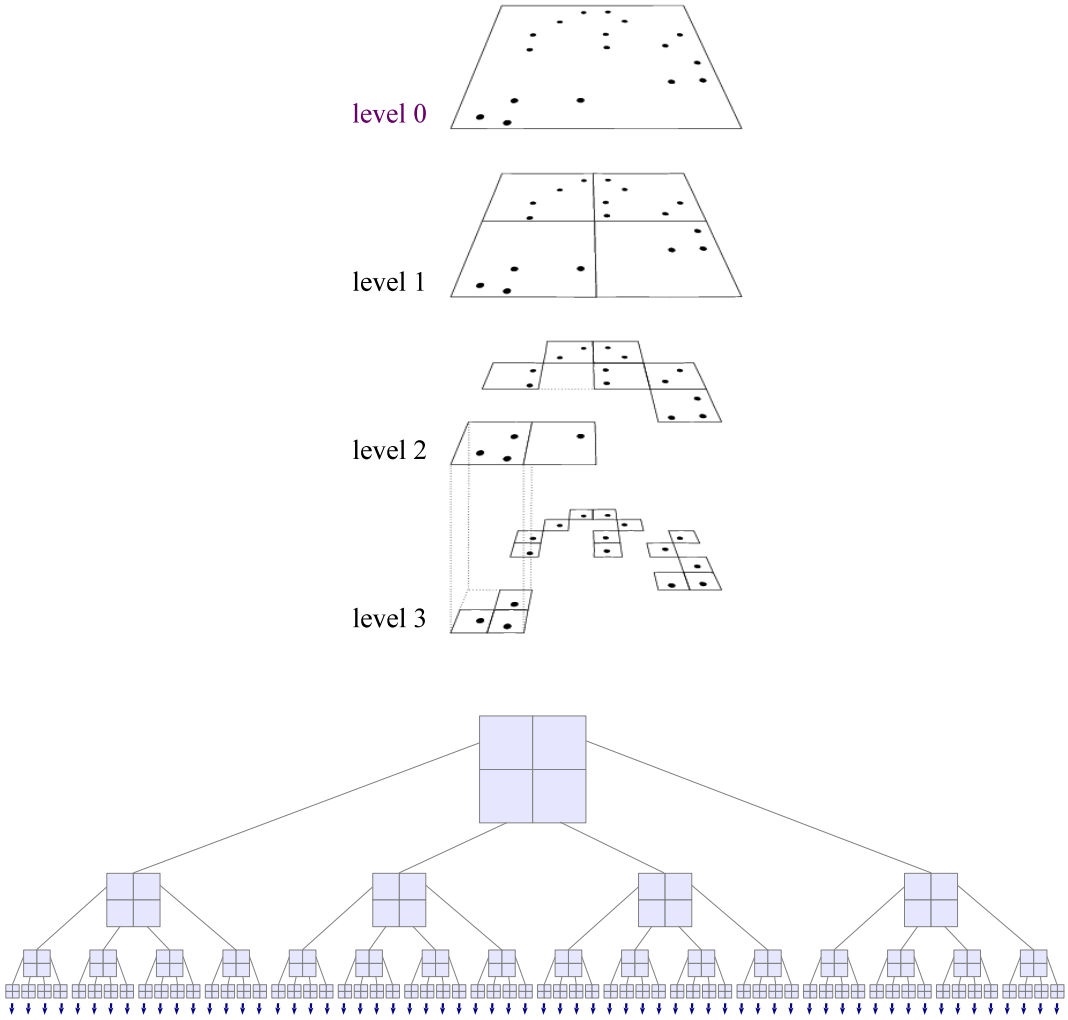
$$\theta \simeq \frac{\langle|\mathbf{x}_i - \mathbf{s}|\rangle}{|\mathbf{y}|} \simeq \frac{l}{y} \ll 1, \tag{4.9}$$

where l is the radius of the group.

4.2 Hierarchical grouping

Tree algorithms are based on a hierarchical grouping of the particles, and for each group, one then pre-computes the multipole moments for later use in approximations of the force due to distant groups. Usually, the hierarchy of groups is organized with the help of a tree-like data structure, hence the name "tree algorithms".

There are different strategies for defining the groups. In the popular Barnes & Hut oct-tree, one starts out with a cube that contains all the particles. This cube is then subdivided into 8 sub-cubes of half the size in each spatial dimension. One continues with this refinement recursively until each subnode contains only a single particle. Empty nodes (sub-cubes) need not be stored. Here is schematic sketch how this can look like in two dimensions (where one has a 'quad-tree'):



The sketch at the bottom shows the topological organization of the tree.

- We note that the oct-tree is not the only possible grouping strategy. Sometimes also so-called kd-trees, or other binary trees are used where subdivisions are done along alternating spatial axes.

- An important property of such hierarchical, tree-based groupings is that they are geometrically fully flexible and adjust to any clustering state the particles may have. They are hence automatically adaptive.

- Also, there is no significant slow-down when severe clustering starts.

- The simplest way to construct the hierarchical grouping is to sequentially insert particles into the tree, and then to compute the multipole moments recursively.

4.3 Tree walk

The force calculation with the tree then proceeds by *walking the tree*. Starting at the root node, one checks for every node whether the opening angle under which it is seen is smaller than a prescribed tolerance angle θ_c. If this is the case, the multipole expansion of the node can be accepted, and the corresponding partial force is evaluated and added to an accumulation of the total force. The tree walk along this branch of the tree can then be stopped. Otherwise, one must open the tree node and consider all its sub-nodes in turn.

The resulting force is then approximate in nature by construction, but the overall size of the error can be conveniently controlled by the tolerance opening angle θ_c. If one makes this smaller, more nodes will have to be opened. This will make the residual force errors smaller, but at the price of a higher computational cost. In the limit of $\theta_c \to 0$ one gets back to the expensive direct summation force.

Cost of the tree-based force computations

How do we expect the total cost of the tree algorithm to scale with particle number N? For simplicity, let's consider a sphere of size R containing N particles that are approximately homogeneously distributed. The mean particle spacing of these particles will then be

$$d = \left[\frac{(4\pi/3)R^3}{N}\right]^{1/3}. \tag{4.10}$$

We now want to estimate the number of nodes that we need for calculating the force on a central particle in the middle of the sphere. We can identify the computational cost with the number of interaction terms that are needed. Since the used nodes must tessellate the sphere, their number can be estimated as

$$N_{\rm nodes} = \int_d^R \frac{4\pi r^2\,{\rm d}r}{l^3(r)}, \tag{4.11}$$

where $l(r)$ is the expected node size at distance r, and d is the characteristic distance of the nearest particle. Since we expect the nodes to be close to their maximum allowed size, we can set $l \simeq \theta_c r$. We then obtain

$$N_{\rm nodes} = \frac{4\pi}{\theta_c^3}\ln\frac{R}{d} \propto \frac{\ln N}{\theta_c^3}. \tag{4.12}$$

The total computational cost for a calculation of the forces for all particles is therefore expected to scale as $\mathcal{O}(N \ln N)$. This is a very significant improvement compared with the N^2-scaling of direct summation.

We may also try to estimate the expected typical force errors. If we keep only monopoles, the error in the force per unit mass from one node should roughly be of order the truncation error, i.e. about

$$\Delta F_{\rm node} \sim \frac{GM_{\rm node}}{r^2}\theta^2. \tag{4.13}$$

The errors from multipole nodes will add up in quadrature, hence

$$(\Delta F_{\mathrm{tot}})^2 \sim N_{\mathrm{node}}(\Delta F_{\mathrm{node}})^2 = N_{\mathrm{node}} \left(\frac{G M_{\mathrm{node}}}{r^2} \theta^2 \right)^2 \propto \frac{\theta^4}{N_{\mathrm{node}}} \propto \theta^7. \tag{4.14}$$

The force error for a scheme with monopoles only therefore scales as $(\Delta F_{\mathrm{tot}}) \propto \theta^{3.5}$, roughly inversely as the invested computational cost.