N-Body models of collisionless systems

In a gravitational system one solves a simpler, alternative system of ordinary differential equations for the mean potential Φ rather than solving directly the Vlasov-Poission equation. The resulting trajectories in phase space should coarsely sample *f* as long as the relaxation time of the N-Body model is long compare to the duration of the simulation. However a single N-Body experiment with a computer will only represent a noisy representation of *f* because no ensemble averaging is performed (one should run many experiments and then average out the resulting forces to get trajectories in the mean potential)

$$\ddot{\mathbf{x}}_i = -\nabla_i \Phi(\mathbf{r}_i),$$

$$\Phi(\mathbf{r}) = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{r}-\mathbf{r}_j)^2 + \varepsilon^2]^{1/2}}$$

The gravitational softening ε is introduced for both computational efficiency and to enforce numerical robustness (=smoothness) of the model for the collisionless fluid. To avoid correlation between particles

$$\langle v^2 \rangle \gg \frac{Gm}{\varepsilon}$$

Note also that the mass of the macro particle used to discretize the model drops out of the equation of motion (no self-force). The trajectory of a macro-particle should thus "trace" the trajectory of real physical particles if the mean potential is sampled well enough to be as close as possible to the actual mean potential. This means again we need large N!

Calculating dynamics of N-Body systems

1st - Compute right side of equation of motion, i.e. gravitational forces --> this gives us accelerations

2nd - Integrate the equation of motion in time so that we update velocities and then positions of particles ---> trajectories in phase space under mean potential, equivalent to evolve the distribution function in time, in turn equivalent to solve the Vlasov-Poisson system!

Let us first focus on step 1. There are various methods to compute forces, one exact and all the others approximate. The exact method is the *direct summation.* One just solves the softened force equation:

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{r}_i - \mathbf{r}_j)^2 + \varepsilon^2]^{3/2}} (\mathbf{r}_i - \mathbf{r}_j)$$

For a system of N particles this means for each of the N particles we have to compute a sum with N partial forces, which is as expensive as $O(N^2)$.
Special purpose hardware, such as large clusters of graphics processing units (GPUs), today can compute up to 10^7 particles in ~ 1 yr. But this means computing ~ 10^{10} particles would already need as much as ~ 1 million years!

Need to turn to approximate methods. These usually reduce to direct summation at short distances but "smooth out" the force using some coarser representation of the density field or potential at large enough distance (or even do this always)

In practice the choice is dictated by a compromise between accuracy of the force calculation and speed of computation (*computational efficiency*). Choice will be application dependent.

Note that accuracy should not be pushed too much since there are inevitable *truncation errors* (eg discretization errors and finite timestep choice for time integration, to be discussed) and *round-off errors of computers (=error due to truncation in floating point number representation)*

Among the methods:

(1) Particle mesh (PM)
(2)Fourier transform-based solvers of Poisson
(3) Iterative solvers of Poission on a grid (multigrid)
(4) Tree-based methods (hierarchical multipole methods)
(5) Hybrid methods, such as TreePM or P^3M

Particle mesh (PM) technique

Underlying idea: construct an auxiliary mesh on which forces are computed by computing the potential

Note: the mesh introduces *a second discretization* procedure after that introduced by super-particles.

The technique involves four steps:

1. Construction of a density field ρ from particle distribution.

This effectively smooths out the discrete representation of the collisionless fluid (reduces N as wanted)

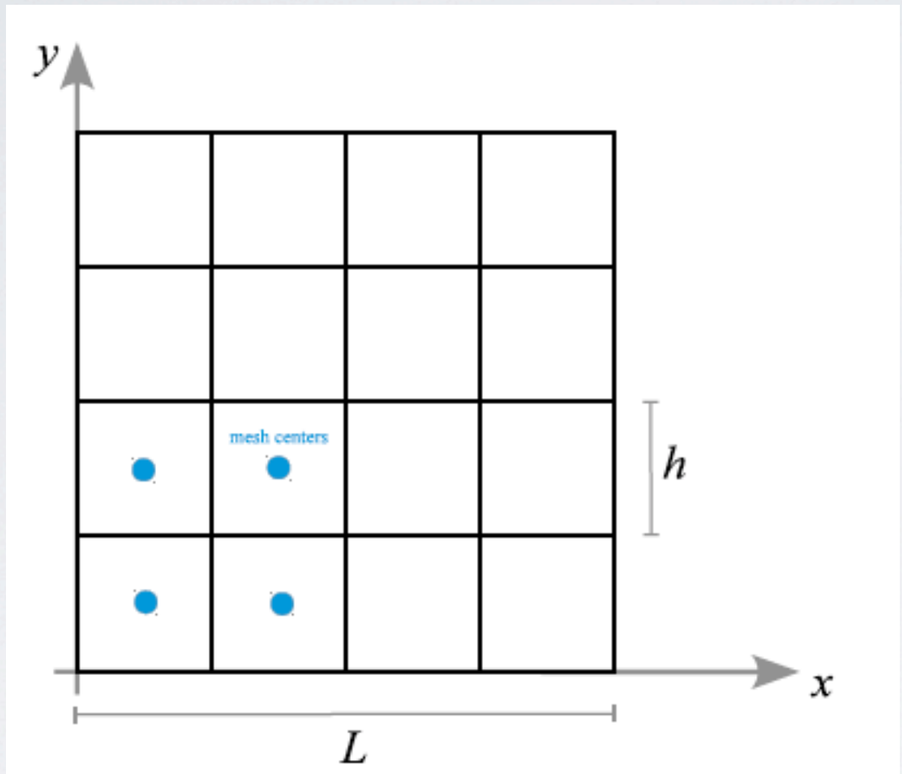
But note this is a smoothing of the super-particle model not of the original collisionless system (eg galaxy) hence it does not eliminate the discretization errors of the particle representation rather can introduce new ones!

2. Computation of the potential on the mesh by solving the Poisson equation with the density field ρ mapped from particle onto the mesh

3. Computation of the force field from the potential (on the mesh)

4. Calculation of the forces at the original particle positions (interpolation step)

Mass assignment procedure



Lay down N_g grid cells that surround particles. For simplicity we assume cubical cells (squares in 2D) with uniform spacing $h=L/N_g$ (*uniform grid*). Labels cells with integer index $\boldsymbol{p}=(p_x,p_y,p_z)$, running from *0 to* N_g, corresponding to cell center position $\boldsymbol{r}_p$,

Now we need to define how to "spread" the mass of particles across cells, which is accomplished by a *shape function* $S(\boldsymbol{x})$ appropriately normalized:

$$\int S(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1.$$

We then need define *the assignment function* $\boldsymbol{W_p(x_i)}$ which expresses the fraction of mass of particle *i* with coordinate $\boldsymbol{x}_i$ that falls into a cell with integer index $\boldsymbol{p}$

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int_{\mathbf{x_p}-\frac{h}{2}}^{\mathbf{x_p}+\frac{h}{2}} S(\mathbf{x}_i - \mathbf{x_p})\,\mathrm{d}\mathbf{x}.$$

Note that integration extends over the cubical cell **p** with size *h*
Then a mathematical step to reformulate for arbitrary integration boundaries using the top-hat Π function:

$$\Pi(\mathbf{x}) = \begin{cases} 1 \text{ for } |\mathbf{x}| \leq \frac{1}{2}, \\ 0 \text{ otherwise,} \end{cases}$$

So that one writes the assignment function *W* as the convolution of **Π** with *S:*

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x} - \mathbf{x}_{\mathbf{p}}}{h}\right) S(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}})\, d\mathbf{x}.$$

The full density in grid cell **p** will be given by the sum of the fractional mass contributions of particles to cell p, which thus depends on the choice of *S* through ***W****:*

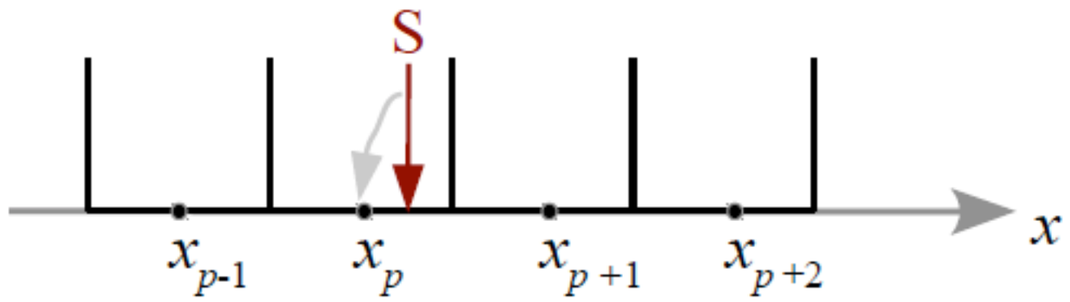
$$\rho_{\mathbf{p}} = \frac{1}{h^3} \sum_{i=1}^{N} m_i W_{\mathbf{p}}(\mathbf{x}_i).$$

The different choice of *S* leads to different mapping between particle distribution and density on the mesh.
The more particle masses are "spread" over more cells via a more complex shape function the smoother is the density function that results, as measured by the continuity of first and second derivatives.

Choices of shape functions *S*

The most trivial choice is $S(\mathbf{x}) = \delta(\mathbf{x})$, which leads *to the nearest grid point mass assignment*, essentially meaning that the entire mass of a particle is assigned only to the cell in which it is embedded. The coordinate of the particle is shifted to that of the center of mass of the cell:

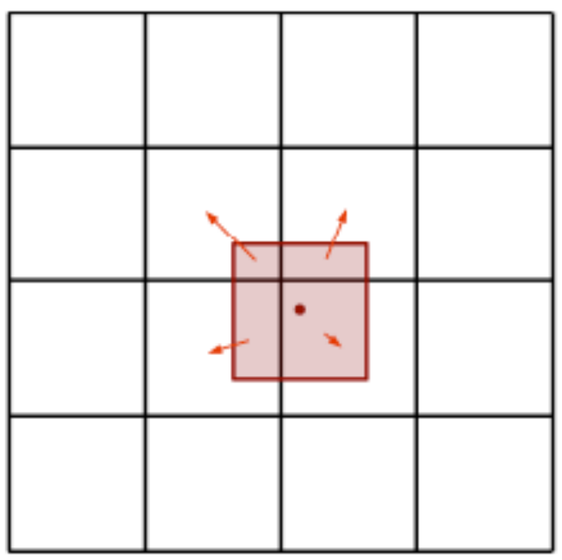
$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x}-\mathbf{x}_{\mathbf{p}}}{h}\right) \delta(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}})\,\mathrm{d}\mathbf{x} = \Pi\left(\frac{\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}}{h}\right)$$



This method is easy but noisy as the assigned density changes abruptly as particle cross cells, leading to a noisy, discontinuous force (recall force obtained from potential obtained from density). It is especially bad for clustered mass distribution (common case in astro and cosmology)

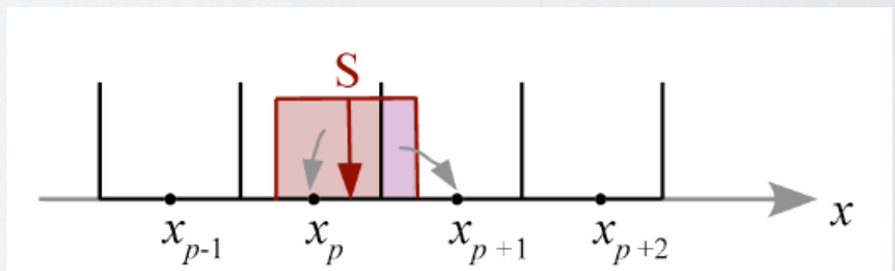
A less noisy mass assignment method (force continous, its first derivative discontinous) is the cloud-in-cell method (CIC).

Underlying idea: embed particles in a cubical cloud (square in 2D), centered on the particle's position, that can overlap with more than one cell. Mass is then redistributed over the cells with non-zero overlap by computing $W_{\mathbf{p}}$, which then yields the "fractional mass weight" associated to each cell, namely what fraction of the particle's mass is attributed to a given cell. $W_{\mathbf{p}}$ computes the weights by computing the fraction of cloud's area that overlaps w/cell **p**



2D CIC
(4 weights
to be computed)

1D CIC
(2 weights
to be computed)



In 3D 8 weights have to computed (mass spread over 8 cells)

Mathematically, one uses following shape function:

$$S(\mathbf{x}) = \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right),$$

resulting in the following mass assignment function:

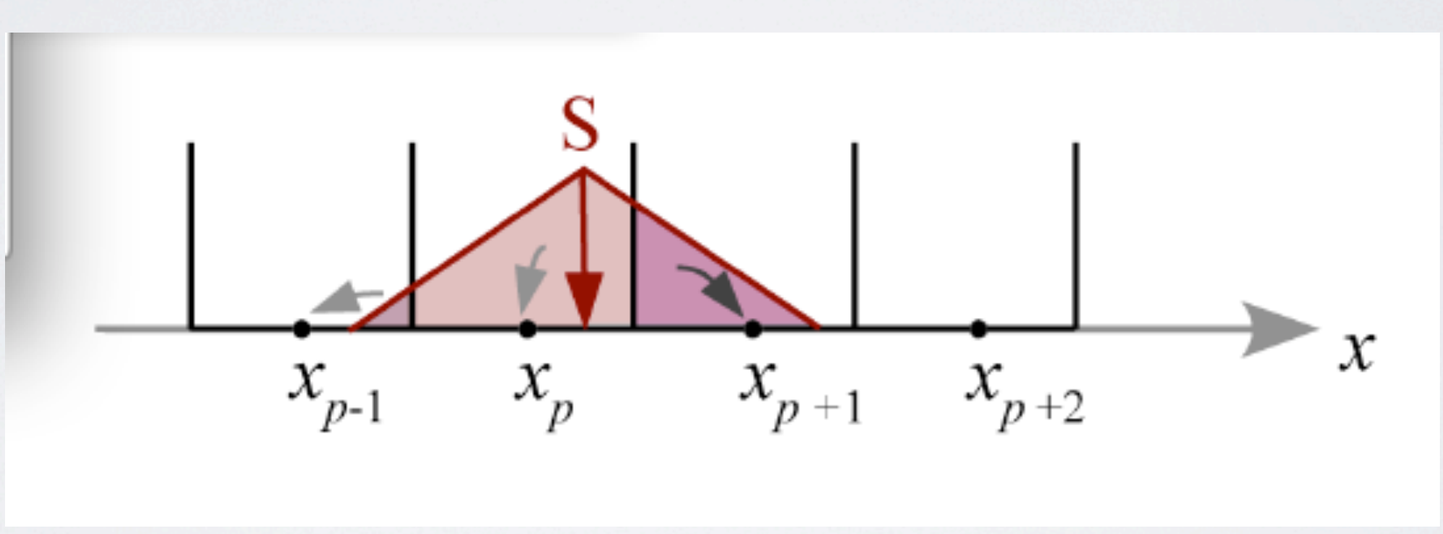
$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x}-\mathbf{x}_{\mathbf{p}}}{h}\right)\frac{1}{h^3}\Pi\left(\frac{\mathbf{x}_i-\mathbf{x}_{\mathbf{p}}}{h}\right)\mathrm{d}\mathbf{x},$$

which only has a non-zero (constant) integrand if the "cubical cloud" of the particle and the cubical mesh cell overlap

Note that CIC is naturally computationally more expensive than NGP because it has to interpolate (=compute fractional weights) across 2^d cells (*d* being the dimension) while NGP uses only the initial cell in which a particle is embedded

Higher-order mass assignment methods can enforce continuity of the derivative of the force by smoothing across an even larger number of cells than CIC, but they become progressively more computationally expensive because more fractional weighs have to be computed

One example is the triangular shaped cloud (TIC), which spreads mass over 3^d cells:



Triangle is defined to have total base length 2h. Assignment function *W* computes area of overlap of triangle with cell

Solving the Poisson equation

$$\nabla^2 \Phi = 4\pi G \rho ,$$

Once the density has been mapped onto the mesh the Poisson equation can be solved with two main classes of methods:

(a) Fourier techniques (convolution methods).
Here map to k-space, find solution in k-space and then map back to real space. Formally natural for periodic boundaries (natural situation in cosmological structure formation) but can be re-cast also for non-periodic boundaries.
Methods requires N^2 operations (N number of grid points) can be reduced to ~ NlogN using Fast Fourier Transform (FFT)

(b)Iterative methods (=find potential using a sequence of approximations with increasing accuracy convergence satisfactory).
Various methods of different accuracy, cost and convergence properties (Jacobi, Gauss-Seidel, multigrid..).

Provided the potential is found the acceleration/force can be determined by finite-differencing on the grid

$$\mathbf{a} = -\nabla\Phi.$$

In one direction the simplest estimate of the force on mesh would be:

$$a_x^{(i,j,k)} = -\frac{\Phi^{(i+1,j,k)} - \Phi^{(i-1,j,k)}}{2h},$$

This goes as $O(h^2)$. If one wants a more accurate determination then can account for more distant cells in the finite-difference estimate of the potential. For example:

$$a_x^{(i,j,k)} = -\frac{1}{2h}\left\{\frac{4}{3}\left[\Phi^{(i+1,j,k)} - \Phi^{(i-1,j,k)}\right] - \frac{1}{6}\left[\Phi^{(i+2,j,k)} - \Phi^{(i-2,j,k)}\right]\right\}$$

which now goes at $O(h^4)$. Same procedure can be followed for force computation in other directions y and z on cartesian mesh. Choice dictated by compromise between accuracy and speed of computation. In final step force on mesh interpolated back to particles (will cover briefly later)

Fourier techniques

These are powerful to solve PDEs whose solutions can be expressed as convolutions of more than one function (normally two) in real space. Then one maps to k-space and uses mathematical properties of Fourier transform. Formally periodic (wave-like solutions) but there are methods to adapt to non-periodic problems.
Numerically can be expensive but there are methods (eg FFT) that speed up calculation a lot by reducing the number of operations exploiting dependencies between terms in a series

Consider again Poisson equation

$$\nabla^2 \Phi = 4\pi G \rho .$$

Simple general "ansatz" can be written by noting that potential at a point **x** must come from superpositions of contributions from individual mass elements (infinitesimally spaced if distribution is continuous), with the contribution of each mass element having the form of a potential from a point mass:

$$\Phi(\mathbf{x}) = -\int G \frac{\rho(\mathbf{x}')\,d\mathbf{x}'}{|\mathbf{x}-\mathbf{x}'|} .$$

This is recognized as a convolution integral of the type:

$$\boldsymbol{\Phi}(\mathbf{x}) = \int g(\mathbf{x}-\mathbf{x}')\,\boldsymbol{\rho}(\mathbf{x}')\,\mathrm{d}\mathbf{x}',$$

where $g(\mathbf{x}) = -\dfrac{G}{|\mathbf{x}|}$ is the Green's function of newtonian gravity

The convolution may be formally written as $\Phi = g * \rho$
Then using the *convolution theorem*, which states that the Fourier transform of the convolution of two functions is the product of their Fourier transforms, one can write:

$$\mathscr{F}(f \star g) = \mathscr{F}(f) \cdot \mathscr{F}(g),$$

where the curl *F* above denotes the Fourier transform. We have reduced the convolution integral into the product of two functions in Fourier space. The potential in k-space can then be determined by:

$$\hat{\boldsymbol{\Phi}}(\mathbf{k}) = \hat{g}(\mathbf{k}) \cdot \hat{\boldsymbol{\rho}}(\mathbf{k})$$

and one maps back into real space potential with $\boldsymbol{\Phi} = \mathscr{F}^{-1}\left[\mathscr{F}(g) \cdot \mathscr{F}(\boldsymbol{\rho})\right]$