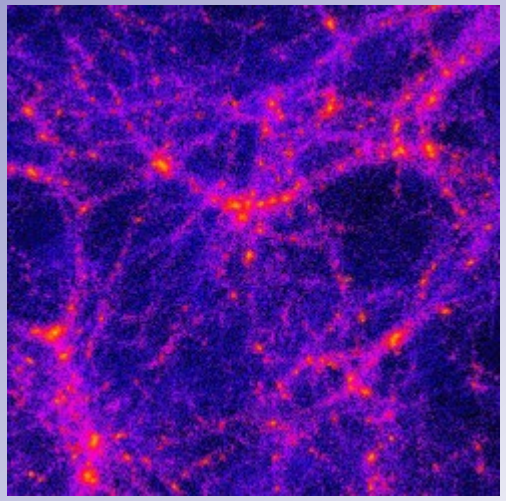
N-Body: The context

- Astrophysical problems with many (>2 !) bodies, with no symmetry able to simplify the treatment: star clusters, globular clusters, dynamical study of the Galaxy, asteroid dynamics, solar sistem formation

- Astrophysical problems where the gravitational evolution of a density field is important. It must be possible to sample the field with particles (Large Scale Structure of the Universe...)



Why numerical simulations in Cosmology

- The problem: structures grow via gravitational evolution of density fluctuations with respect to an homogeneous background

- Dynamics is described by the Vlasov-Poisson equations:

$$\nabla^2 \Phi(\vec{x},t) = 4\pi a^2(t)\,\delta(\vec{x},t) \quad \textit{Poisson}$$

$$\frac{\partial f(\vec{x},\vec{p},t)}{\partial t} + \frac{\vec{p}}{m a^3(t)} \cdot \nabla f(\vec{x},\vec{p},t) - m \nabla \Phi(\vec{x},t) \cdot \frac{\partial f(\vec{x},\vec{p},t)}{\partial \vec{p}} = 0 \quad \textit{Vlasov}$$

Distribution function:

f(x,p,t) describes the dynamics of mass elements in the phase space (6D) initially determined by P(k)

- No general analitic solution; semi-analitical appsoximations hold in weakly non-linear regime.
---> we need numerical solutions

Euler-Poisson equations

$\rho_b(t)$:average density of the Universe at time t

$\delta(x,t)$: density contrast $\delta(x,t)=[\rho(x,t)-\rho_b(t)]/\rho_b(t)$

Φ : peculiar gravitational potential

f(x,p,t) : mass elements distribution function

If we impose f(x,p,t) to be a single-valued function of positions (no multistream)we get Euler-Poisson equations:

$$\frac{\partial \delta(\vec{x},t)}{\partial t}+\nabla\cdot\vec{v}(\vec{x},t)\delta(\vec{x},t)=0$$ Continuity

$$\frac{\partial \vec{v}(\vec{x},t)}{\partial t}+2\frac{\dot{a}(t)}{a(t)}\vec{v}(\vec{x},t)+(\vec{v}(\vec{x},t)\cdot\nabla)\vec{v}(\vec{x},t)=-\frac{\partial \Phi(\vec{x},t)}{\partial t}$$ Euler

$$\nabla^2\Phi(\vec{x},t)=4\pi a^2(t)\delta(\vec{x},t)$$ Poisson

Note: we use *density contrast* not global density: equations describe the evolution of *structures* not that of the Universe. Cosmology contained in the espansion factor a(t)

Simulation of gravitational interaction

- Simpler method: a set of massive points interact gravitationally: Newton's force:

$$\vec{F}_i = \sum_j \frac{G m_i m_j \left(\vec{r}_i - \vec{r}_j\right)}{\left|\vec{r}_i - \vec{r}_j\right|^3}$$

- All particles interact with each other one ***(direct code)***

- Discretization of time: first, all forces are evaluated, then, all particles are shifted by one *time-step*.

- Boundary conditions naturally **void**

Direct N-Body codes: problems

- We need an accurate control over the time step to avoid integration errors

- We need a sophisticated integration algorithm to get accuracy, control over the numerical error and symplectic integration (basically, time-reversible solution of the Hamilton equations)

- The nearer particles are, the smaller the time step must be. If the time-step is too long, particles can be expelled on hyperbolic trajectories: ***collisionality***.

- Needed computing time proprortional to the square of the number of particles (***computational complexity***):

$$T \propto N^2$$

Softening

- To overcome the collisionality problem, a *softening parameter* is introduced. It changes the shape of the force:

$$\vec{F}_i = \sum_j \frac{G m_i m_j (\vec{r}_i - \vec{r}_j)}{\left(|\vec{r}_i - \vec{r}_j| + \epsilon^2\right)^{3/2}}$$

- On the softening scale, the force *is not Newtonian any more,* and it is not even conservative (!)
- Tests: simulations results are statistically equivalent to that of an unsoftened calculation on a scale

$$l \propto K\epsilon, K \simeq 1.5$$

- Various forms are possible for the softening (K varies!)

Regularization; variable time-steps

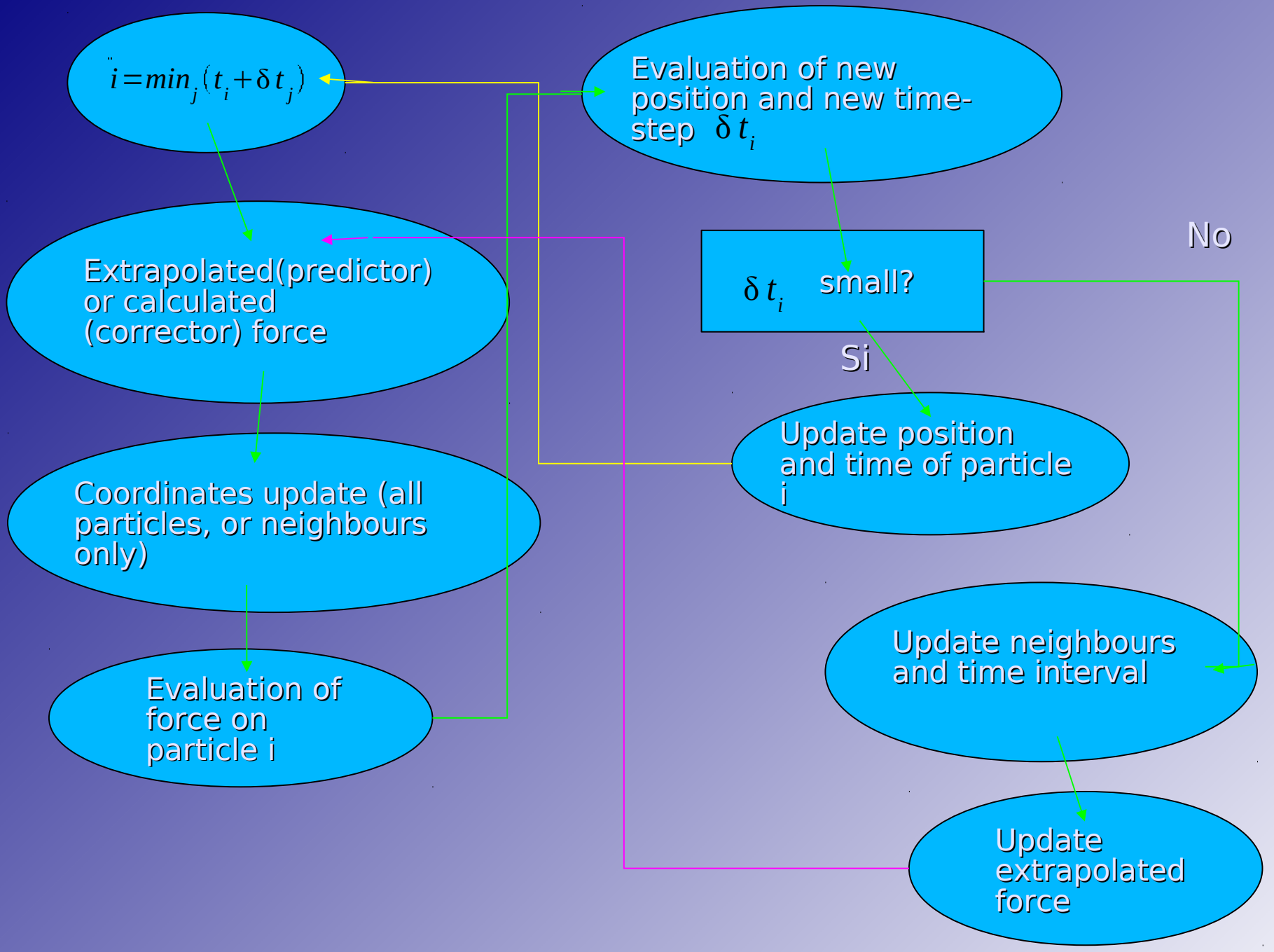
- It is not always possible to ignore small-scale behaviour of simulated systems. Example: globular clusters.

- *Regularization*: semi-analitical treatment of 2 or 3 body encounters.

- Finite-differences integration technique have been introduced (Ahmad-Cohen 1973). Variable time-step for each particle. Predictor-corrector methods to be able to use them.

- "Standard" scheme for a direct (or ***PP*** ,Particle-Particle) code

PP code scheme



N-Body in Cosmology

Dark matter --> *non collisional cold fluid.*
N-body "particles" represent a fluid element : we need the largest possible number of them

- Particles must *sample* the fluid:
mass assignment schemes

- ***Softening needed!*** (a fluid has NO two body encountes)
or, an equivalent technique

- ***Comoving coordinates:*** we follow the evolution of *density perturbation* (Vlasov-Poisson) – not of the density field.

- Gravitational evolution equations coefficients now depend on time, through the **scale factor *a(t)***

- ***a(t) contains the cosmological model and all of the needed General Relativity***
-
- Direct integration of the field (PP) is numerically too expensive!

Particle-Mesh codes

- Grid techniques and FFT used (Poisson eq. Solved in the Fouries space, integration in the configuration space) NO “direct” collisionality

- Boundary conditions are naturally periodic

- Gravitational potential and forces are computed on a grid and interpolated to the particles; density field is recomputed assigning the mass using the new particle positions

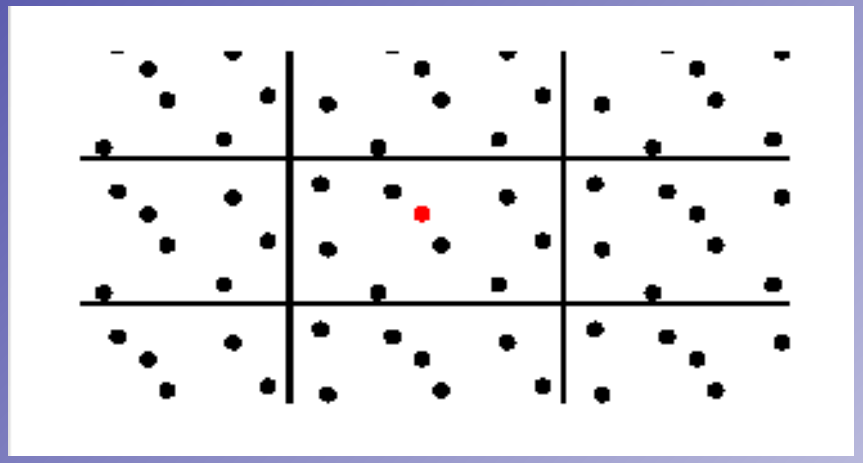
- Integrator: second order leap frog with time-dependent cofficients

- Time dependend coefficients do contain *all* the cosmology (model and its dynamics)

- Computing time dominated by FFT cost; computational complexity:

$$T \propto N \log_2 N$$

Mass assignment scheme and interpolation



- Simplest schemes:

- NGP (Nearest Grid Point): mass of particle given to the cell where it belongs. Discountinuous field.

- CIC (Cloud In Cell): a particle occupy a region of space (“cloud”) of size R, mass is assigned proportionally to the amount of particle volume that falls inside each cell.
Continuous fiels, first derivative discountinous. Tipically R=L, 8 cells

- TCS (Triangular Shaped Cloud): assignment is not linear, but weighted so that the first derivative is continuous. R=1.5L (27 cells).

- Force is interpolated with a scheme that is the inverse of that used for the mass assignment, to avoid *self-forces*.

Green functions

- To solve Poisson eq. in Fourier space, we need to calculate the Green function of the Laplacian operator in such a space

- "Simple-man green function":

$$G = \frac{cost}{k^2}$$

- This function refers to the operator working on R^3; but the derivation is discrete (in cells) and the quantity assignment scheme must be considered; the simplest Green function results:

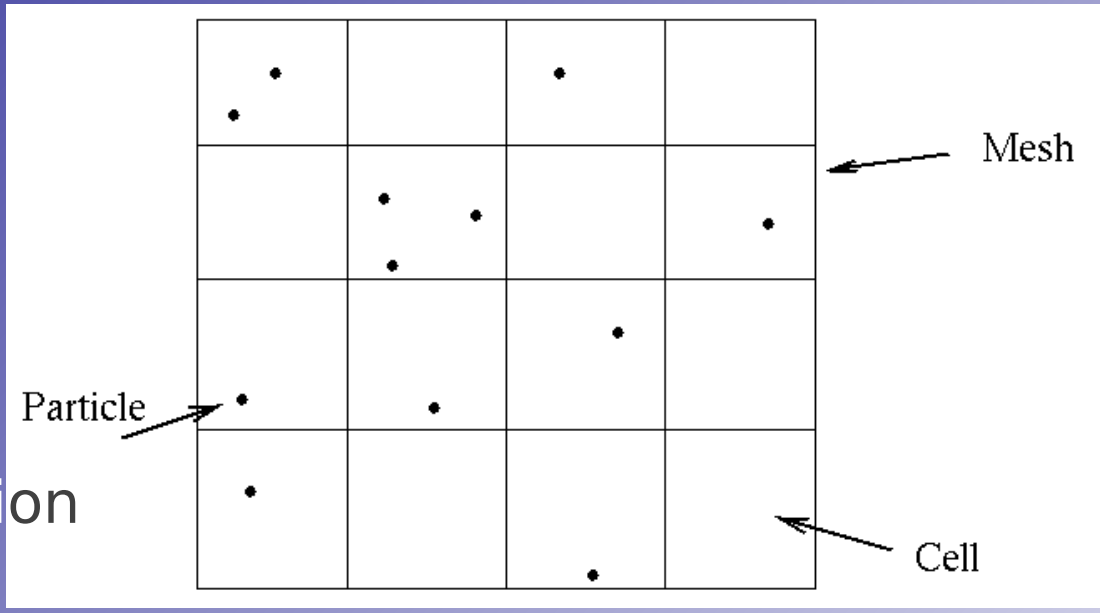
$$G = \frac{-\pi}{L^2} \frac{1}{\sin^2 2\pi k_x / L + \sin^2 2\pi k_y / L + \sin^2 2\pi k_z / L}$$

(Hockney & Eastwood, "Computer simulations using particles)

1st STEP: assign densities to the mesh from particle positions

$$\rho_m = \frac{1}{h^3} \sum_i m_i W(\mathbf{x}_i - \mathbf{x}_m)$$

$W(\mathbf{x}_m - \mathbf{x}_i)$: weighting function



2nd STEP: solve the Poisson equation in Fourier space

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

Solution of the Poisson eq. with $g(\mathbf{x}) = -\mathrm{G}/|\mathbf{x}|$: Green's function of the Laplacian

Use FFT to compute $\hat{\Phi}(\mathbf{k}) = \hat{g}(\mathbf{k})\,\hat{\rho}(\mathbf{k})$

Transform back to compute $\Phi(\mathbf{x})$

3rd STEP: compute the force on the grid:

$$\mathbf{f}(\mathbf{x}) = -\nabla\Phi(\mathbf{x})$$

Use a finite differentiation:

$$f^{(x)}_{i,j,k} = -\frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h}$$

4th STEP: interpolate back forces to particle positions, using the same weighting scheme:

$$\mathbf{f}(\mathbf{x}_i) = \sum_m W(\mathbf{x}_i - \mathbf{x}_m)\mathbf{f}_m$$

5th STEP: update particle positions and velocities

E.g. using the “leapfrog” scheme to integrate $\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$

Kick-Drift-Kick

$$\mathbf{v}_{n+1/2} = \mathbf{v}_n + \mathbf{f}(\mathbf{x}_n)\Delta t/2$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_{n+1/2}\Delta t$$
$$\mathbf{v}_{n+1} = \mathbf{v}_{n+1/2} + \mathbf{f}(\mathbf{x}_{n+1})\Delta t/2$$

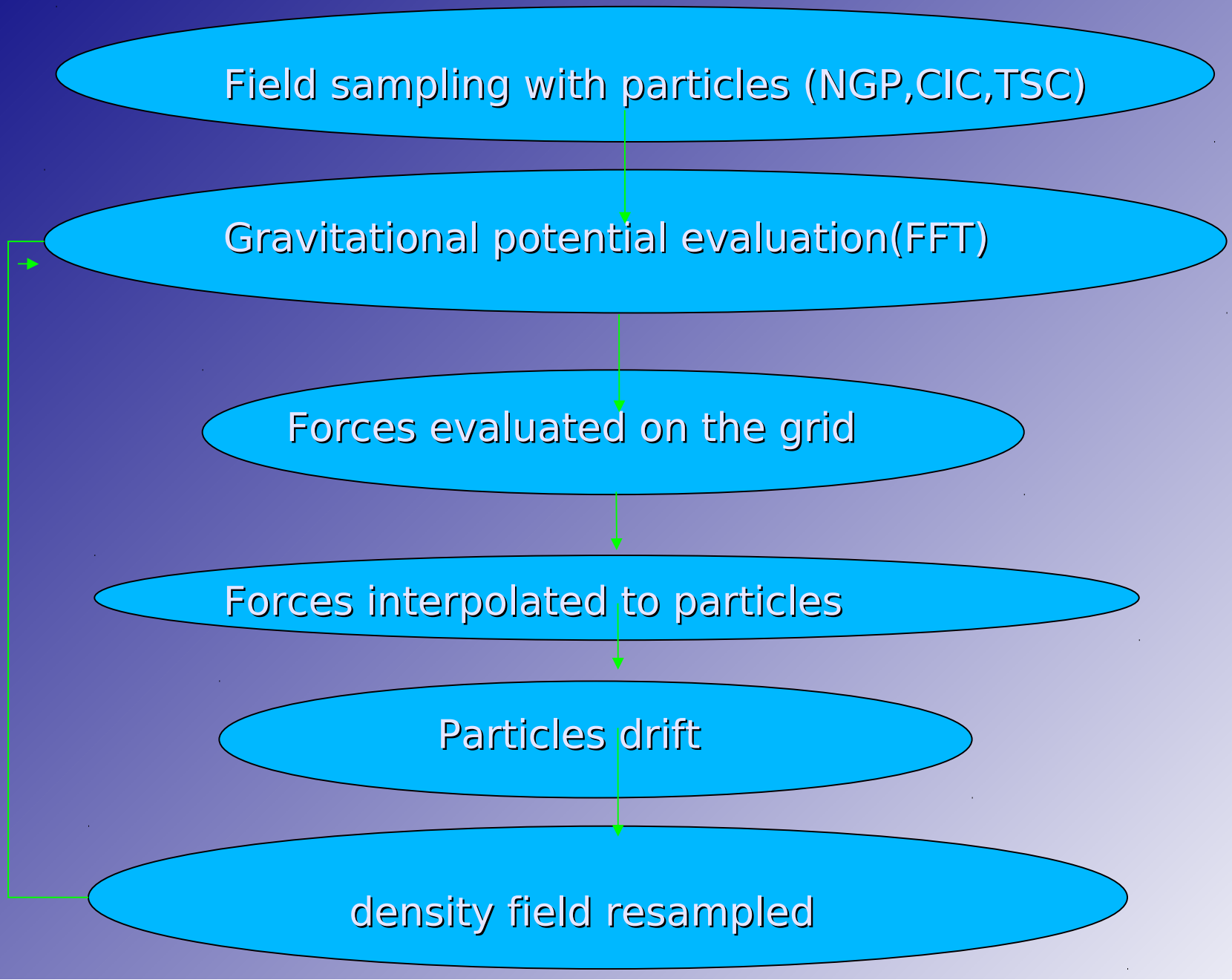
Drift-Kick-Drift

$$\mathbf{x}_{n+1/2} = \mathbf{x}_n + \mathbf{v}_n\Delta t/2$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{f}(\mathbf{x}_{n+1/2})\Delta t$$
$$\mathbf{x}_{n+1} = \mathbf{x}_{n+1/2} + \mathbf{v}_{n+1}\Delta t/2$$

$$\Delta t = \alpha\sqrt{\epsilon/|\mathbf{a}|}$$

$\alpha \approx 0.1$

Particle-Mesh code scheme



PM codes: problems

- Resolution L of simulation is the grid resolution; i reliable results on scales $\sim 3L$
- "Second order collisionality": from grid discretness (depends on the chosen mass assignment scheme)
- Integration error of order O(L^2)??
- Energy conservation is not perfect (Layzier-Irvine equation)
- These codes are NOT ready, without important modifications, to problems that have non-periodic boundary conditions(e.g.: solar system study; simulation of a single galaxy or galaxy cluster)
- Replica problem: objects MUST NOT go non-linear on scales larger than 1/10 of the box-size

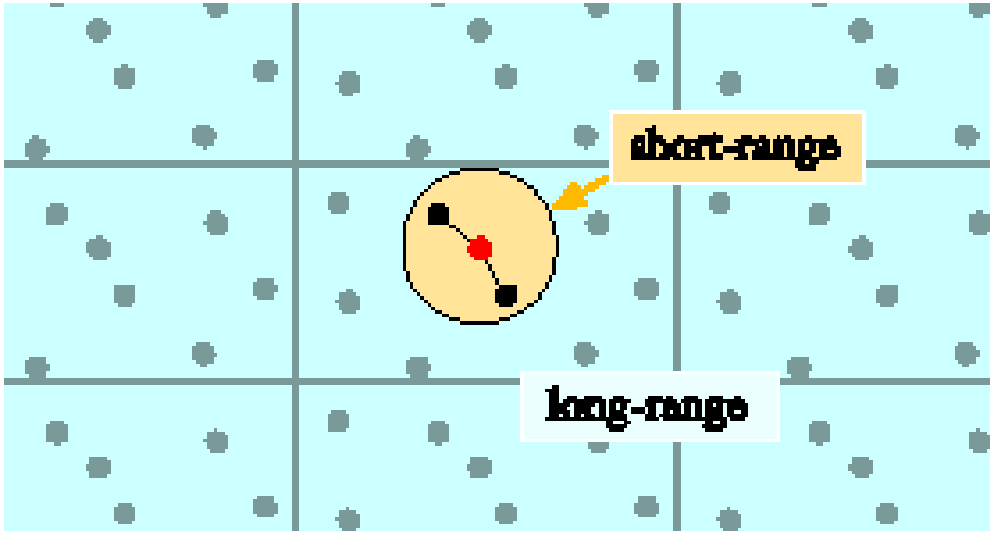
Derived codes

- P3M: PM at large scales, PP at small scales.

PROS: PP force resolution but much faster
CONS: softening <--> collisionality; difficult to implement; PM/PP interface; mass resolution << force resolution

- APM: adaptive PM, iterates the grid construction procedure in overdense regions

PROS: force resolution similar to a PP but speed of a PM
CONS: see P3M



- AP3M: mix of the above techniques

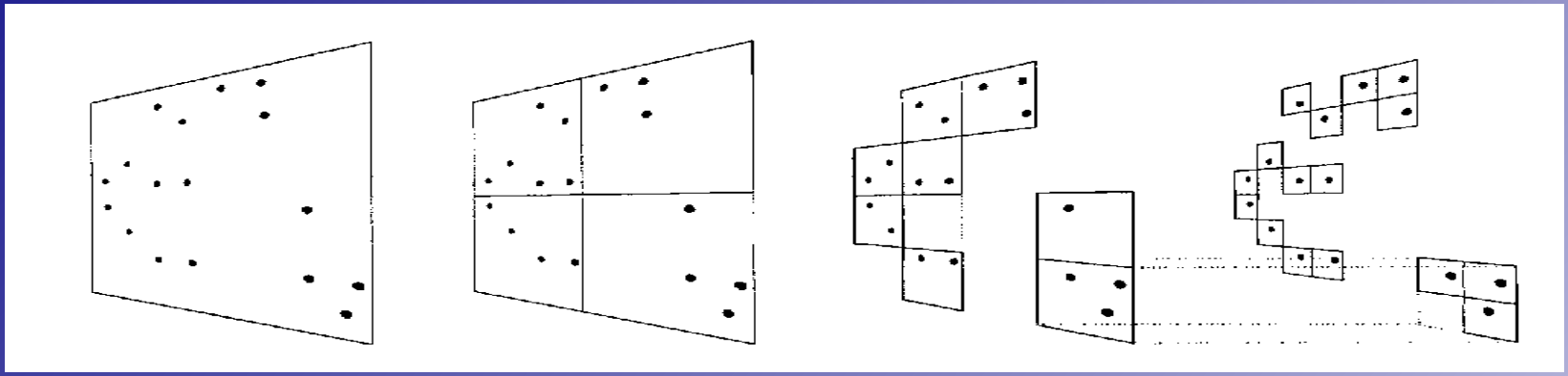
More modern codes

- Treecode: ~ multipole development of mass, tree-ordered data. Allows for multi-mass techniques
  PROS: PP resolution but NlogN complexity. Can easily simulate single objects
  CONS: softening <--> collisionality; difficult to implement **and parallelize**; not periodic; integration error control more problematic

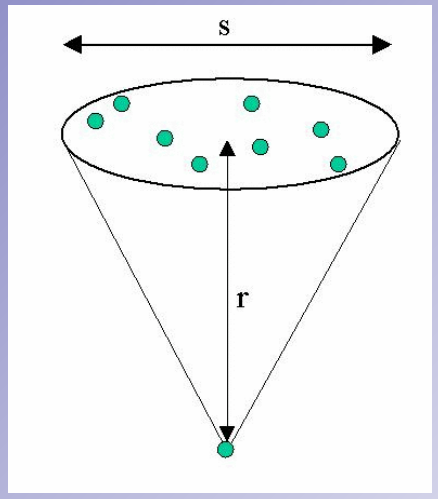
- "ART": (A. Klypin) adaptive PM using tree techniques, multimass, without grid iteration
  PROS: PP resolution; can naturally simulate an high-resolution object embedded in a low-resolution environment
  CONS: difficult to implement; slowest than [A]P3M; no SPH / HYDRO (but: S. Kravtsov work)

Treecode

- Space divided in cells iteratively, until each cell contains one only particle



Each particle do interact with the *center of mass* of first level cells, unless their size is seen under an angle larger than the *tolerance parameter* $\theta = s/r$.
In this case, the interaction is with center of masses of second level cells and so on.
Neighbouring particles interacts directly (PP)



Adaptive Refinement Tree

- PM code at the “zeroth” level

- If in a cell a given density threshold is passed, it is divided in eight parts, then:
Poisson equation is transformed into a diffusion equation:

$$\nabla^2\Phi=\rho \;\rightarrow\; \frac{\partial\Phi}{\partial\tau}=\nabla^2\Phi-\rho$$

An initial solution “relaxes” to the equilibrium solution – solution of the Poisson equation - for

$$\tau\rightarrow\infty$$

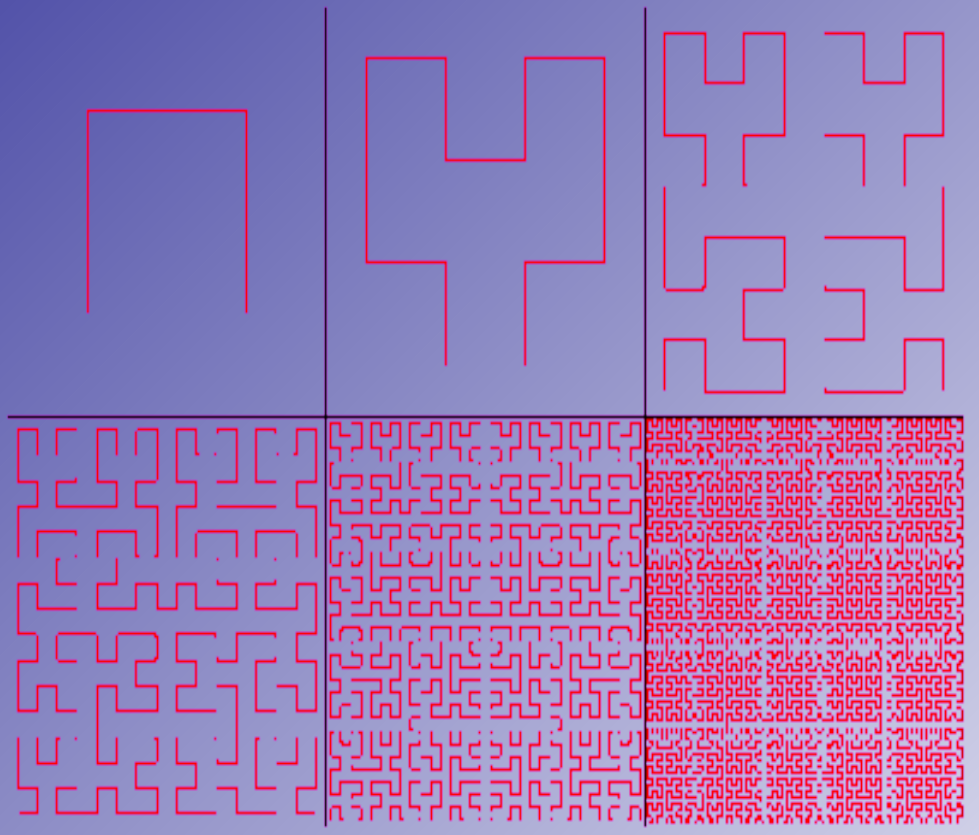
- The initial solution is taken from the previous refinement level – at the first level is the PM solution. Procedure is iterated.

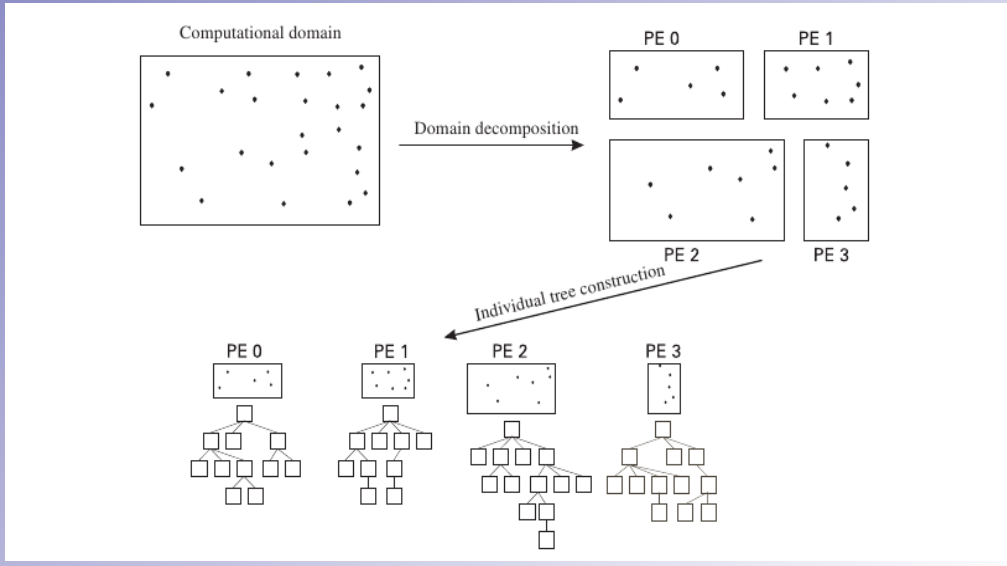
Parallel codes

- Large dynamical lrange: Gpc → < 1 kpc
- ....very large number of computational elemtens needed: supercomputers...

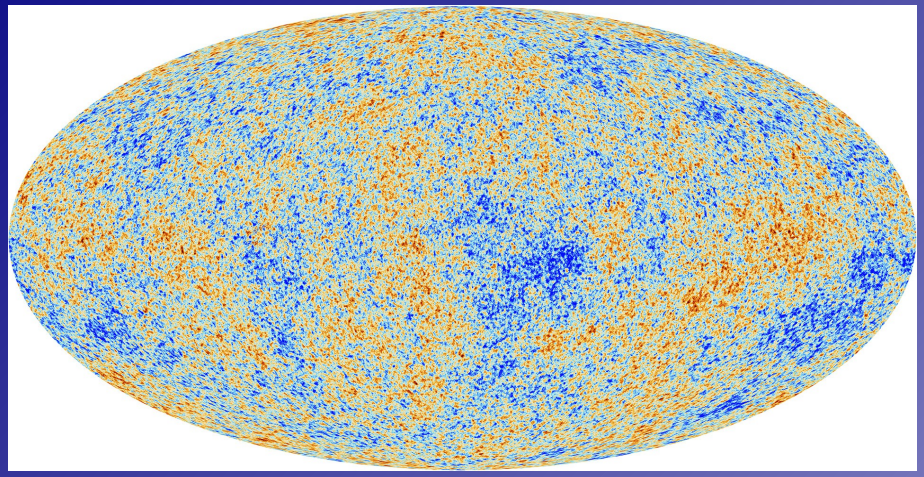
- Large computing power available with **massively parallel computers** (many CPU with many cores, now with accelerators)
- Numerical codes must distribute computations among various tasks, threads, kernels...

- Many numerical problems: load balancing, memory, parallelism level, code architecture

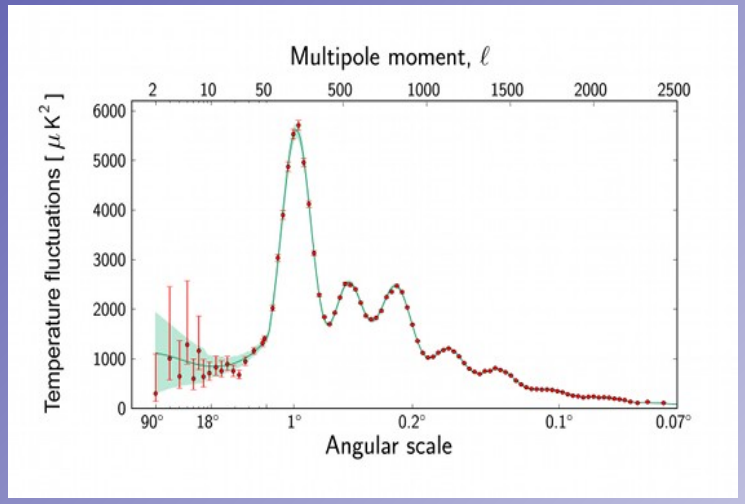




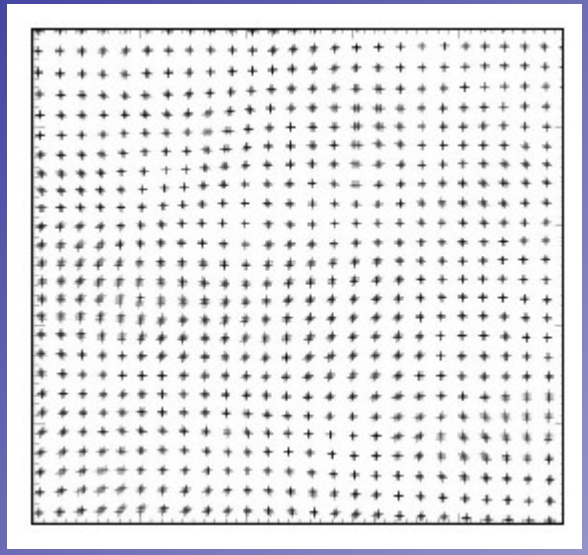
Initial conditions



Plank satellite, CMB



$$P(k) = T^2(k) P_i(k)$$



Zeldovich approximation:

$$\vec{x}(t) = \vec{q} + D(t)\vec{S}(\vec{q})$$

$$D(t) = \frac{\delta(\vec{x},t)}{\delta_0(\vec{x},t_0)}$$