PiTP Summer School 2009

Plan for my lectures

Volker Springel

Lecture 1  Basics of collisionless dynamics and the N-body approach

Lecture 2  Gravitational solvers suitable for collisionless dynamics, parallelization

Lecture 3  More parallelization, Introduction to smoothed particle hydrodynamics

Lecture 4  Algorithmic aspects of SPH, caveats, applications

Lecture 5  Comparison of SPH to finite volume methods, Moving-mesh hydrodynamics
The N-body approach to collisionless dynamics

BASIC MONTE-CARLO IDEA

Poisson-Vlasov System

\[
\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \cdot v + \frac{\partial f}{\partial v} \cdot \left( -\frac{\partial \Phi}{\partial x} \right) = 0
\]

\[
\nabla^2 \Phi(x, t) = 4\pi G \int f(x, v, t) \, dv
\]

N-body System

\[
\ddot{x}_i = -\nabla \Phi(x_i)
\]

\[
\Phi(x) = -G \sum_{j=1}^{N} \frac{m_j}{\left[ (x - x_j)^2 + \epsilon^2 \right]^{1/2}}
\]
Several questions come up when we try to use the N-body approach for collisionless simulations

- How do we compute the gravitational forces efficiently and accurately?
- How do we integrate the orbital equations in time?
- How do we generate appropriate initial conditions?
- How do we parallelize the simulation?

Note: The naïve computation of the forces is an $N^2$-task.

\[ \ddot{x}_i = -\nabla_i \Phi(x_i) \]

\[ \Phi(x) = -G \sum_{j=1}^{N} \frac{m_j}{\left[ (x - x_j)^2 + \epsilon^2 \right]^{1/2}} \]
Initial conditions generation
In special cases, the distribution function for static solutions of the CBE can be constructed analytically.

An integral of motion \( I = I(x(t), v(t)) \) is constant along orbits, i.e.:
\[
\frac{dI}{dt} = 0
\]

Then \( I \) is a solution of the CBE.

**Jeans theorem:** Steady-state solutions of the CBE only depend on integrals of motion.

For a spherical mass distribution, a DF that only depends on energy can be constructed with Eddington's formula.

**Example:**

**Hernquist halo:**
\[
\rho(r) = \frac{M}{2\pi} \frac{a}{r(r + a)^3}
\]
\[
f(E) = \frac{1}{\sqrt{2}(2\pi)^{3/2}(GMa)^3/2} \frac{\sqrt{e}}{(1 - e)^2} \left[ (1 - 2e)(8e^2 - 8e - 3) + \frac{3\sin^{-1}(\sqrt{e})}{\sqrt{e}(1 - e)} \right]
\]

where:
\[
e = -\frac{aE}{GM} \quad \quad E = \frac{v^2}{2} + \Phi
\]
Construction of compound disk galaxies that are in dynamical equilibrium

STRUCTURAL PROPERTIES OF MODEL GALAXIES

Components:

- Dark halo (Hernquist profile matched to NFW halo)
- Stellar disk (exponential)
- Stellar bulge
- Gaseous disk (exponential)
- Central supermassive black hole

One approach: Compute the exact gravitational potential for the axisymmetric mass distribution and solve the Jeans equations
The first step in constructing an isolated galaxy model is the specification of the density structure of all mass components.

**DENSITY DISTRIBUTIONS OF DARK MATTER AND STARS IN BULGE AND DISK**

**Dark matter:**

\[ \rho_{dm}(r) = \frac{M_{dm}}{2\pi} \frac{a}{r(r + a)^3} \]

Hernquist or NFW profile

**Stars in the disk:**

\[ \Sigma_*(r) = \frac{M_*}{2\pi h^2} \exp\left(-\frac{r}{h}\right) \]

"Isothermal sheet" with exponential profile

\[ \rho_*(R, z) = \frac{M_*}{4\pi z_0 h^2} \text{sech}^2 \left( \frac{z}{2 z_0} \right) \exp\left(-\frac{R}{h}\right) \]

**Stars in the bulge:**

\[ \rho_b(r) = \frac{M_b}{2\pi} \frac{b}{r(r + b)^3} \]

**Gas in the disk:**

\[ \Sigma_{gas}(r) = \frac{M_{gas}}{2\pi h^2} \exp\left(-\frac{r}{h}\right) \]

Vertical structure given by hydrostatic equilibrium. Depends on the equation of state of the gas.

\[ -\frac{1}{\rho_g} \frac{\partial P}{\partial z} - \frac{\partial \Phi}{\partial z} = 0 \]
Solving the Jeans equations allows the construction of dynamically stable disk galaxy models

MOMENT EQUATIONS FOR THE VELOCITY STRUCTURE

We assume that the velocity distribution function of dark matter and stars can be approximated everywhere by a triaxial Gaussian.

Further, we assume axisymmetry, and that the distribution function depends only on $E$ and $L_z$.

Then cross-moments vanish:

$$\langle v_R v_z \rangle = \langle v_z v_\phi \rangle = \langle v_R v_\phi \rangle = 0$$

$$\langle v_R \rangle = \langle v_z \rangle = 0$$

The radial and vertical moments are given by:

$$\langle v_z^2 \rangle = \langle v_R^2 \rangle = \frac{1}{\rho} \int_{-\infty}^{\infty} \rho(z',R) \frac{\partial \Phi}{\partial z'} \, dz'$$

The azimuthal dispersion fulfills a separate equation:

$$\langle v_{\phi}^2 \rangle = \langle v_R^2 \rangle + \frac{R}{\rho} \frac{\partial}{\partial R} \left( \rho \langle v_R^2 \rangle \right) + v_c^2$$

Circular velocity: $v_c^2 \equiv R \frac{\partial \Phi}{\partial R}$

A remaining freedom lies in the azimuthal streaming $\langle v_\phi \rangle$, which is not determined by the above assumptions. For the dark matter, it can be set to zero, or to a value corresponding to a prescribed spin.

Note: For the stellar disk, we instead use the epicycle theory to relate radial and vertical dispersions.
The famous **merger hypothesis** conjectures that tidal features around galaxies occur in collisions which ultimately produce spheroidals.

TOOMRE & TOOMRE (1972 !)

**Restricted three-body simulations**

A model for the interaction of M51 and NGC 5195
More important than particle number is physical insight and intuition

DAVIS, EFSTATHIOU, FRENK & WHITE (1985 !)

$32^3$ particles – the first generation of CDM simulations
In modern simulations, the same dark matter autocorrelation function is measured, but more accurately

**DARK MATTER TWO-POINT FUNCTION**

![Graph showing the dark matter two-point function with data points and a line. The graph is labeled Millennium, $z = 0.00$. It plots the autocorrelation function $\xi(r)$ on the y-axis against the comoving distance $r [h^{-1} \text{ Mpc}]$ on the x-axis. The graph includes a logarithmic scale for both axes, with values ranging from $10^0$ to $10^{10}$ particles on the y-axis and from $0.01$ to $100.00$ on the x-axis.]
The initial conditions for cosmic structure formation are directly observable

THE MICROWAVE SKY

If the initial fluctuations are a Gaussian random field, we only need to know the power spectrum and the cosmological parameters to describe the ICs.

**DIFFERENT PROBES OF THE MASS POWER SPECTRUM**

(figure from Max Tegmark)
The linear theory power spectrum can be computed accurately.
To determine the power spectrum amplitude, we normalize the spectrum to observations of clustering (usually galaxy clusters)

**FILTERED DENSITY FIELD AND THE NORMALIZATION OF THE POWER SPECTRUM**

The filtered density field:

\[ \sigma^2(M, z) = D^2(z) \int_0^\infty \frac{dk}{2\pi^2} k^2 P(k) \left[ \frac{3j_1(kR)}{kR} \right]^2 \]

**Observational input:**

\[ \sigma_8 = 0.74 - 0.9 \quad R = 8 \, h^{-1}\text{Mpc} \]

Extrapolate back to the starting redshift with the growth factor \( D(z) \)
This depends on cosmology.

\[ \text{fluctuation spectrum of initial conditions fully specified.} \]

Aside:

\[ P(k) \propto k^n \rightarrow \sigma^2(M) \propto M^{-(n+3)/3} \]

- Close to the critical slope, halos on very different mass scales form nearly simultaneously
- The multiplicity function of halos becomes essentially flat
To create a realization of the perturbation spectrum, a model for an unperturbed density field is needed.

GLASS OR CARTESIAN GRID

For CDM, the initial velocity dispersion is negligibly small.

But there is a mean streaming velocity, which we need to imprint in initial conditions.
Using the Zeldovich approximation, density fluctuations are converted to displacements of the unperturbed particle load

**SETTING INITIAL DISPLACEMENTS AND VELOCITIES**

**Particle displacements:**

\[ \mathbf{d}_i(t) = \mathbf{x}_i(t) - \mathbf{q}_i \]

**Density change due to displacements:**

\[ \rho(\mathbf{x}) = \frac{\rho_0}{\partial_x} = \frac{\rho_0}{\delta_{ij} + \frac{\partial \mathbf{d}}{\partial \mathbf{q}}} \]

**For small displacements:**

\[ \left| \delta_{ij} + \frac{\partial \mathbf{d}}{\partial \mathbf{q}} \right| \simeq 1 + \nabla_{\mathbf{q}} \cdot \mathbf{d} \]

**Resulting density contrast:**

\[ \delta(\mathbf{x}) = \frac{\rho(\mathbf{x}) - \rho_0}{\rho} = -\nabla_{\mathbf{q}} \cdot \mathbf{d} \]

**During linear growth:**

\[ \delta(t) = D(t)\delta_0 \]

\[ \mathbf{d}(t) = D(t)\mathbf{d}_0 \]

\[ \mathbf{x} = \dot{\mathbf{d}} = \frac{\dot{a}}{a} \frac{dD}{da} \mathbf{d}_0 = \frac{\dot{a}}{a} \dot{D} \frac{dD}{da} \mathbf{d} \]

**Particle velocities:**

\[ \dot{x} = H(a)f(\Omega)\mathbf{d} \quad f(\Omega) = \frac{d\ln D}{d\ln a} \simeq \Omega^{0.6} \]

**Displacement field:**

\[ \nabla^2 \phi = \delta \quad \mathbf{d} = -\nabla \phi \]

**Fourier realization:**

\[ \phi_k = -\frac{1}{k^2}\delta_k \quad \mathbf{d}_k = -ik\phi_k = \frac{ik}{k^2}\delta_k \quad \mathbf{d}_k = -\nabla \phi = \sum_k \frac{ik\delta_k}{k^2} \exp(ik\mathbf{x}) \]

Note: Particles move on straight lines in the Zeldovich approximation.
One can assign random amplitudes and phases for individual modes in Fourier space

**GENERATING THE FLUCTUATIONS IN K-SPACE**

For each mode, draw a random phase, and an amplitude from a Rayleigh distribution.

\[
\delta_k = B_k \exp^{i\phi_k}
\]

\[
\langle \delta_k^2 \rangle = P(k)
\]

Simulation box sampled with \(N^2\) points
Calculating gravitational forces
Direct summation calculates the gravitational field exactly

FORCE ACCURACY IN COLLISIONLESS SIMULATIONS

Direct summation approach:
\[ \ddot{x}_i = -\nabla_i \Phi(x_i) \]

\[ \Phi(x) = -G \sum_{j=1}^{N} \frac{m_j}{\left[ (x - x_j)^2 + \epsilon^2 \right]^{1/2}} \]

Are approximate force calculations sufficient?

Yes, provided the force errors are random and small enough.

Since the N-body force field is noisy anyway, small random errors will only insignificantly reduce the relaxation time.

Systematic errors in the force, or error correlations are however very problematic.
Cosmological N-body simulations have grown rapidly in size over the last three decades.

- Computers double their speed every 18 months (Moore's law).
- N-body simulations have doubled their size every 16-17 months.
- Recently, growth has accelerated further. The Millennium Run should have become possible in 2010 – we it was done in 2004. It took ~350000 CPU hours, about a month on 512 cores.

\[ N = 400 \times 10^{0.215(\text{Year} - 1975)} \]
The particle mesh (PM) force calculation
The particle-mesh method

Poisson's equation can be solved in real-space by a convolution of the density field with a Green's function.

\[ \Phi(x) = \int g(x - x') \rho(x) \, dx' \]

Example for vacuum boundaries:

\[ \Phi(x) = -G \int \frac{\rho(x)}{|x - x'|} \, dx' \quad g(x) = -\frac{G}{|x|} \]

In Fourier-space, the convolution becomes a simple multiplication!

\[ \hat{\Phi}(k) = \hat{g}(k) \cdot \hat{\rho}(k) \]

→ Solve the potential in these steps:

1. FFT forward of the density field
2. Multiplication with the Green's function
3. FFT backwards to obtain potential

The four steps of the PM algorithm

(a) Density assignment
(b) Computation of the potential
(c) Determination of the force field
(d) Assignment of forces to particles
Give particles a “shape” $S(x)$. Then to each mesh cell, we assign the fraction of mass that falls into this cell. The overlap for a cell is given by:

$$W(x_m - x_i) = \int_{x_m - \frac{h}{2}}^{x_m + \frac{h}{2}} S(x' - x_i) \, dx' = \int \Pi \left( \frac{x' - x_m}{h} \right) S(x' - x_i) \, dx'$$

The assignment function is hence the convolution:

$$W(x) = \Pi \left( \frac{x}{h} \right) \ast S(x)$$

where

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

The density on the mesh is then a sum over the contributions of each particle as given by the assignment function:

$$\rho(x_m) = \frac{1}{h^3} \sum_{i=1}^{N} m_i \, W(x_i - x_m)$$
# Commonly used particle shape functions and assignment schemes

<table>
<thead>
<tr>
<th>Name</th>
<th>Shape function $S(x)$</th>
<th># of cells involved</th>
<th>Properties of force</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGP</td>
<td>$\delta(x)$</td>
<td>$1^3 = 1$</td>
<td>piecewise constant in cells</td>
</tr>
<tr>
<td>Nearest grid point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIC</td>
<td>$\frac{1}{h^3} \prod \left( \frac{x}{h} \right) * \delta(x)$</td>
<td>$2^3 = 8$</td>
<td>piecewise linear, continuous</td>
</tr>
<tr>
<td>Clouds in cells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSC</td>
<td>$\frac{1}{h^3} \prod \left( \frac{x}{h} \right) * \frac{1}{h^3} \prod \left( \frac{x}{h} \right)$</td>
<td>$3^3 = 27$</td>
<td>continuous first derivative</td>
</tr>
<tr>
<td>Triangular shaped clouds</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For interpolation of the grid to obtain the forces, the same assignment function needs to be used to ensure momentum conservation. (In the CIC case, this is identical to tri-linear interpolation.)
Finite differencing of the potential to get the force field

Approximate the force field \( \mathbf{f} = - \nabla \Phi \) with finite differencing.

2\textsuperscript{nd} order accurate scheme:

\[
f_{i,j,k}(x) = - \frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h}
\]

4\textsuperscript{th} order accurate scheme:

\[
f_{i,j,k}(x) = - \frac{4}{3} \frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h} + \frac{1}{3} \frac{\Phi_{i+2,j,k} - \Phi_{i-2,j,k}}{4h}
\]

Interpolating the mesh-forces to the particle locations

\[
F(x_i) = \sum_m W(x_i - x_m) f_m
\]

The interpolation kernel needs to be the same one used for mass-assignment to ensure force anti-symmetry.
Advantages and disadvantages of the PM-scheme

**Pros:** SPEED and simplicity

**Cons:**
- Spatial force resolution limited to mesh size.
- Force errors somewhat anisotropic on the scale of the cell size.

*serious problem:*
- Cosmological simulations cluster strongly and have a very large dynamic range.
- Cannot make the PM-mesh fine enough and resolve internal structure of halos as well as large cosmological scales.

We need a method to increase the dynamic range available in the force calculation.
Particle-Particle PM schemes ($P^3M$)

Idea: Supplement the PM force with a direct summation short-range force at the scale of the mesh cells. The particles in cells are linked together by a chaining list.

Offers much higher dynamic range, but becomes slow when clustering sets in.

In $AP^3M$, mesh-refinements are placed on clustered regions

Can avoid clustering slow-down, but has higher complexity and ambiguities in mesh placement

Codes that use $AP^3M$: HYDRA (Couchman)
Iterative Poisson solvers can determine the potential directly on a (hierarchical grid)

Idea: Start with a trial potential and then iteratively relax the solution by updating with a finite difference approximation to the Laplacian.

$$\Phi'_{i,j,k} = \frac{1}{6} \left( \Phi_{i+1,j,k} + \Phi_{i-1,j,k} + \Phi_{i,j+1,k} + \Phi_{i,j-1,k} + \Phi_{i,j,k+1} + \Phi_{i,j,k-1} - 4\pi G h^2 \rho_{i,j,k} \right)$$

This updating eliminates errors on the scale of a few grid cells rapidly, but longer-range fluctuations die out much more slowly.

In multigrid methods, a hierarchy of meshes is used to speed up convergence, resulting in a fast method that allows for locally varying resolution.

Examples for codes that use a real-space Poisson solver:

- ART (Kravtsov)
- MLAPM (Knebe)

On adaptive meshes, sometimes a combination of Fourier techniques and real-space solvers is used.
TREE algorithms
Tree algorithms approximate the force on a point with a multipole expansion

**Idea:** Group distant particles together, and use their multipole expansion.

Only $\sim \log(N)$ force terms per particle.
Tree algorithms

Idea: Use hierarchical multipole expansion to account for distant particle groups

$$\Phi(r) = -G \sum_i \frac{m_i}{|r - x_i|}$$

We expand:

$$\frac{1}{|r - x_i|} = \frac{1}{|(r - s) - (x_i - s)|}$$

for $$|x_i - s| \ll |r - s|$$  \quad y = r - s

and obtain:

$$\frac{1}{|y + s - x_i|} = \frac{1}{|y|} - \frac{y \cdot (s - x_i)}{|y|^3} - \frac{1}{2} \frac{y^T [3(s - x_i)(s - x_i)^T - I (s - x_i)^2]}{|y|^5} y + \ldots$$

the dipole term vanishes when summed over all particles in the group
The multipole moments are computed for each node of the tree

Monopole moment:

\[ M = \sum_i m_i \]

Quadrupole tensor:

\[ Q_{ij} = \sum_k m_k \left[ 3(x_k - s)_i(x_k - s)_j - \delta_{ij}(x_k - s)^2 \right] \]

Resulting potential/force approximation:

\[ \Phi(\mathbf{r}) = -G \left[ \frac{M}{|\mathbf{y}|} + \frac{1}{2} \frac{\mathbf{y}^T \mathbf{Q} \mathbf{y}}{|\mathbf{y}|^5} \right] \]

For a single force evaluation, not \( N \) single-particle forces need to be computed, but only of order \( \log(N) \) multipoles, depending on the opening angle.

- The tree algorithm has no intrinsic restrictions for its dynamic range
- Force accuracy can be conveniently adjusted to desired level
- The speed does depend only very weakly on clustering state
- Geometrically flexible, allowing arbitrary geometries
The fast multipole method (FFM) generalizes the tree algorithm and expands the field symmetrically for each pair of interacting cells.

**Two interacting cells:**

- Very fast
- Manifest momentum conservation

**But:**
- Doesn't work well with individual timesteps
- Difficult to parallelize for distributed memory machines
TreePM force calculation algorithm
Particularly at high redshift, it is expensive to obtain accurate forces with the tree-algorithm

THE TREE-PM FORCE SPLIT

**Periodic peculiar potential**

$$\nabla^2 \phi(x) = 4\pi G [\rho(x) - \bar{\rho}] = 4\pi G \sum_n \sum_i m_i \left[ \tilde{\delta}(x - x_i - nL) - \frac{1}{L^3} \right]$$

**Idea:** Split the potential (of a single particle) in Fourier space into a long-range and a short-range part, and compute them separately with PM and TREE algorithms, respectively.

Poisson equation in Fourier space:

$$\phi_k = -\frac{4\pi G}{k^2} \rho_k \quad (k \neq 0)$$

\[\begin{align*}
\phi_k^{\text{long}} &= \phi_k \exp(-k^2 r_s^2) \\
\phi_k^{\text{short}} &= \phi_k \left[ 1 - \exp(-k^2 r_s^2) \right]
\end{align*}\]

Solve with PM-method
- CIC mass assignment
- FFT
- multiply with kernel
- FFT backwards
- Compute force with 4-point finite difference operator
- Interpolate forces to particle positions

FFT to real space

$$\phi(r) = -\frac{Gm}{r} \text{erfc} \left( \frac{r}{2r_s} \right)$$

Solve in real space with TREE
In the TreePM algorithm, the tree has to be walked only locally.

**PERFORMANCE GAIN DUE TO LOCAL TREE WALK**

\[
\phi(r) = -\frac{Gm}{r} \text{erfc}\left(\frac{r}{2r_s}\right)
\]

Advantages of TreePM include:
- Accurate and fast long-range force
- No force anisotropy
- Speed is largely insensitive to clustering (as for tree algorithm)
- No Ewald correction necessary for periodic boundary conditions

Using zero-padding and a different Greens-Function, the long-range force can also be computed for vacuum boundaries using the FFT. (Implemented in Gadget-2)
Brief comments on time integration
Symplectic integration schemes can be generated by applying the idea of operating splitting to the Hamiltonian

**THE LEAPFROG AS A SYMPLECTIC INTEGRATOR**

Separable Hamiltonian

\[ H = H_{\text{kin}} + H_{\text{pot}} \]

Drift- and Kick-Operators

\[ D(\Delta t) = \exp \left( \int_t^{t+\Delta t} dt \ H_{\text{kin}} \right) = \begin{cases} \mathbf{p}_i & \rightarrow & \mathbf{p}_i \\ \mathbf{x}_i & \rightarrow & \mathbf{x}_i + \frac{\mathbf{p}_i}{m_i} \Delta t \end{cases} \]

\[ K(\Delta t) = \exp \left( \int_t^{t+\Delta t} dt \ H_{\text{pot}} \right) = \begin{cases} \mathbf{x}_i & \rightarrow & \mathbf{x}_i \\ \mathbf{p}_i & \rightarrow & \mathbf{p}_i - \sum_j m_i m_j \frac{\partial \phi(\mathbf{x}_{ij})}{\partial \mathbf{x}_i} \Delta t \end{cases} \]

The drift and kick operators are symplectic transformations of phase-space!

The Leapfrog

Drift-Kick-Drift:

\[ \tilde{U}(\Delta t) = D \left( \frac{\Delta t}{2} \right) K(\Delta t) D \left( \frac{\Delta t}{2} \right) \]

Kick-Drift-Kick:

\[ \tilde{U}(\Delta t) = K \left( \frac{\Delta t}{2} \right) D(\Delta t) K \left( \frac{\Delta t}{2} \right) \]

Hamiltonian of the numerical system:

\[ \tilde{H} = H + H_{\text{err}} \quad H_{\text{err}} = \frac{\Delta t^2}{12} \left\{ H_{\text{kin}}, H_{\text{pot}} \right\} + H_{\text{kin}} + \frac{1}{2} H_{\text{pot}} + \mathcal{O}(\Delta t^3) \]
When an adaptive timestep is used, much of the symplectic advantage is lost.

INTEGRATING THE KEPLER PROBLEM

Going to KDK reduces the error by a factor 4, at the same cost!
For periodic motion with adaptive timesteps, the DKD leapfrog shows more time-asymmetry than the KDK variant.

**LEAPFROG WITH ADAPTIVE TIMESTEP**

**DKD**

- Direction of $\Delta t$:
  - Forwards
  - Backwards

**KDK**

- Direction of $\Delta t$:
  - Forwards
  - Backwards
Collisionless dynamics in an expanding universe is described by a Hamiltonian system.

THE HAMILTONIAN IN COMOVING COORDINATES

Conjugate momentum \( \mathbf{p} = a^2 \mathbf{x'} \)

\[
H(p_1, \ldots, p_n, x_1, \ldots, x_n, t) = \sum_i \frac{p_i^2}{2m_i a(t)^2} + \frac{1}{2} \sum_{ij} m_im_j \phi(x_i - x_j)
\]

Drift- and Kick operators

\[
D(t + \Delta t, t) = \exp \left( \int_t^{t + \Delta t} dt H_{\text{kin}} \right) = \begin{cases}
\mathbf{p}_i &\rightarrow& \mathbf{p}_i \\
\mathbf{x}_i &\rightarrow& \mathbf{x}_i + \frac{\mathbf{p}_i}{m_i} \int_t^{t + \Delta t} \frac{dt}{a^2}
\end{cases}
\]

\[
K(t + \Delta t, t) = \exp \left( \int_t^{t + \Delta t} dt H_{\text{pot}} \right) = \begin{cases}
\mathbf{x}_i &\rightarrow& \mathbf{x}_i \\
\mathbf{p}_i &\rightarrow& \mathbf{p}_i - \sum_j m_im_j \frac{\partial \phi(x_{ij})}{\partial \mathbf{x}_i} \int_t^{t + \Delta t} \frac{dt}{a}
\end{cases}
\]

Choice of timestep

For linear growth, fixed step in \( \log(a) \) appears most appropriate...

\[
\Delta t = \frac{\Delta \log a}{H(a)}
\]

timestep is then a constant fraction of the Hubble time.
The force-split can be used to construct a symplectic integrator where long- and short-range forces are treated independently.

**TIME INTEGRATION FOR LONG AND SHORT-RANGE FORCES**

Separate the potential into a long-range and a short-range part:

\[
H = \sum_i \frac{p_i^2}{2m_i a(t)^2} + \frac{1}{2} \sum_{ij} m_i m_j \varphi_{sr}(x_i - x_j) \frac{1}{a(t)} + \frac{1}{2} \sum_{ij} m_i m_j \varphi_{lr}(x_j - x_j) \frac{1}{a(t)}
\]

The short-range force can then be evolved in a symplectic way on a smaller timestep than the long range force:

\[
\tilde{U}(\Delta t) = K_{lr} \left( \frac{\Delta t}{2} \right) \left[ K_{sr} \left( \frac{\Delta t}{2m} \right) D \left( \frac{\Delta t}{m} \right) K_{sr} \left( \frac{\Delta t}{2m} \right) \right]^m K_{lr} \left( \frac{\Delta t}{2} \right)
\]

\[\text{drift} \quad \Delta t \quad \text{short-range force-kick} \quad \text{long-range force-kick} \quad \text{long-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \quad \text{short-range force-kick} \]
Parallelization:
Domain decomposition
The maximum size of collisionless dark matter simulations with the TreePM algorithm is essentially memory bound.

**MEMORY REQUIREMENTS IN DIFFERENT PARTS OF THE (LEAN) GADGET-2 CODE**

<table>
<thead>
<tr>
<th>Component</th>
<th>Memory Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle Data</td>
<td>44 bytes / particle</td>
</tr>
<tr>
<td>Tree storage</td>
<td>40 bytes / particle</td>
</tr>
<tr>
<td>FFT workspace</td>
<td>24 bytes / mesh-cell</td>
</tr>
</tbody>
</table>

Special code version **Lean-GADGET-II** needs: 84 bytes / particle (Assuming 1.5 mesh-cells/particle)

**Example: Simulation set-up of the Millennium Run**

- **Particle number:** \(2160^3 = 10.077.696.000 = \sim 10^{10}\) particles
- **Boxsize:** \(L = 500 \, h^{-1}\) Mpc
- **Particle mass:** \(m_p = 8.6 \times 10^8 \, h^{-1} M_\odot\)
- **Spatial resolution:** \(5 \, h^{-1} \) kpc
- **Size of FFT:** \(2560^3 = 16.777.216.000 = \sim 17\) billion cells

Minimum memory requirement of simulation code: \(~840\) GByte

We will do \(~200\) billion particles soon – but how can we cope with the data?
The **domain decomposition** distributes particles onto different processors.
The space-filling Hilbert curve is a fractal that fills the square.

Constitution of a flexible domain decomposition with cache benefits

Idea: Order the particles along a space-filling curve.
The space-filling Hilbert curve can be readily generalized to 3D

THE PEANO-HILBERT CURVE
The space-filling Peano-Hilbert is used in GADGET-2 for the domain-decomposition

**SPLITTING UP THE TREE FOR DIFFERENT PROCESSORS**