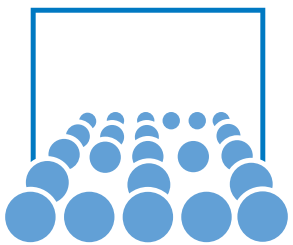
HPC – Algorithms and Applications

Dwarf # 4 – N-Body Methods

Michael Bader

Winter 2012/2013



Part I

N-Body Methods – Implementation

Computational Effort

- to solve: system of ODE

$$\frac{d^2}{dt^2}\vec{r}_i = \frac{\vec{F}_i}{m_i} = \frac{1}{m_i}\sum_{j\neq i}\vec{F}_{ij}$$

- requires forces between all pairs of molecules:

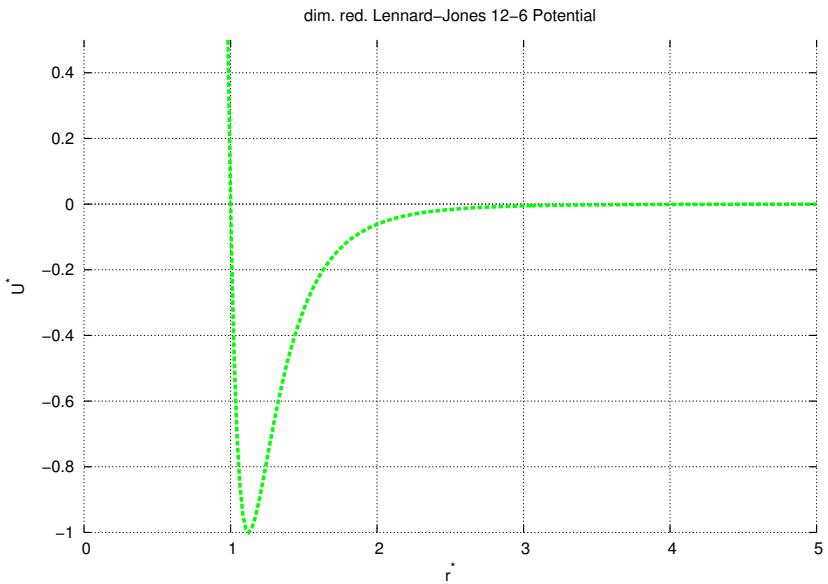
$$\text{“compute } \vec{F}_i = \sum_{j\neq i}\vec{F}_{ij} \text{ for all } i\text{”}$$

- computational effort to compute all forces $\vec{F}_i$ thus $\mathcal{O}(N^2)$
- unfeasible for systems with 10^6 to 10^9 molecules

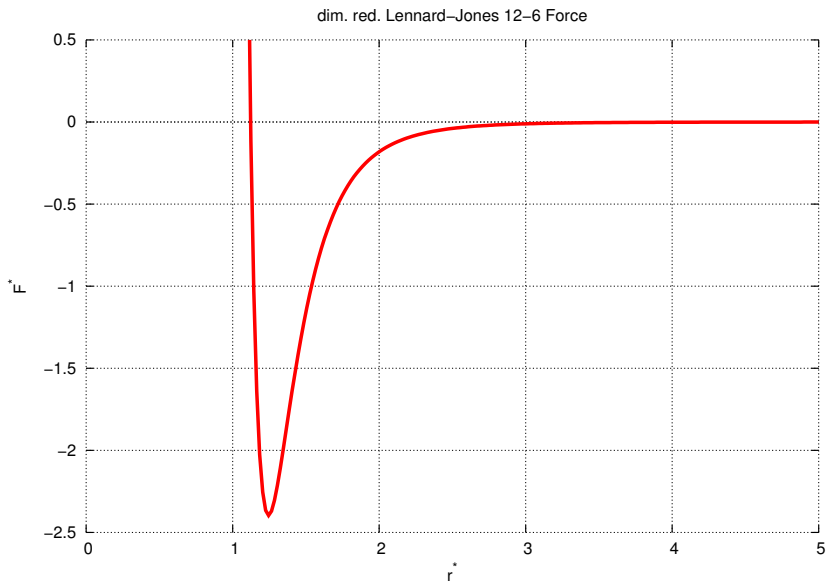
$\Rightarrow$ **How can the computational effort be reduced?**

Short-Range Potentials

- for Lennard-Jones potential, forces decay very quickly for large distances
- idea: neglect forces, if molecules are outside a certain *cut-off range*

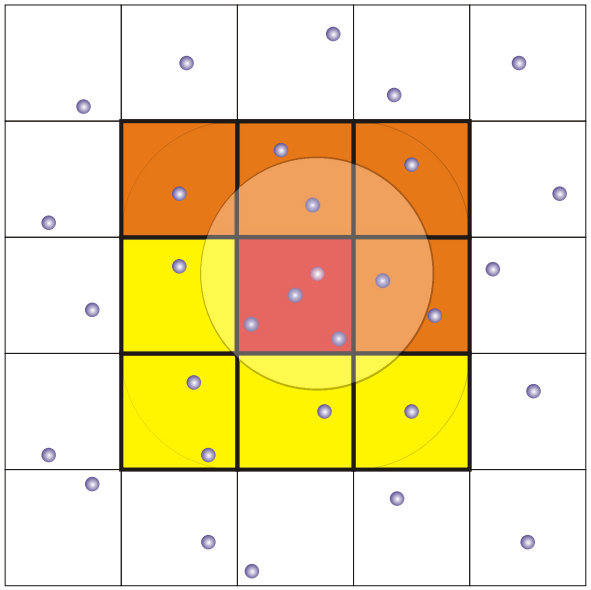


(Force)



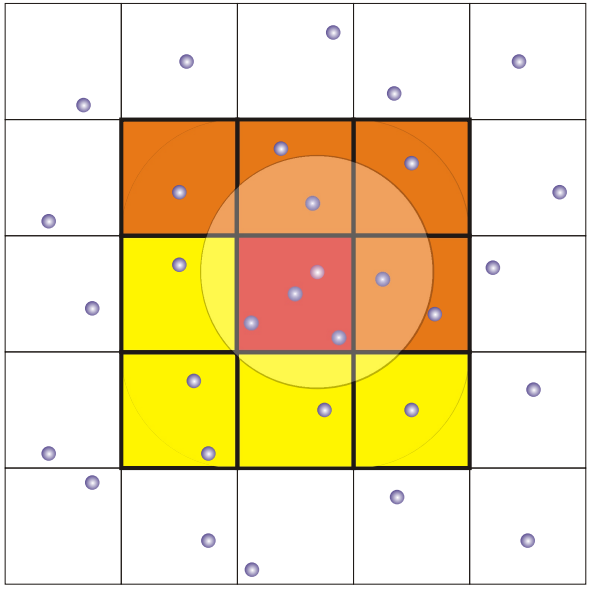
(Potential)

Linked-Cell Algorithm



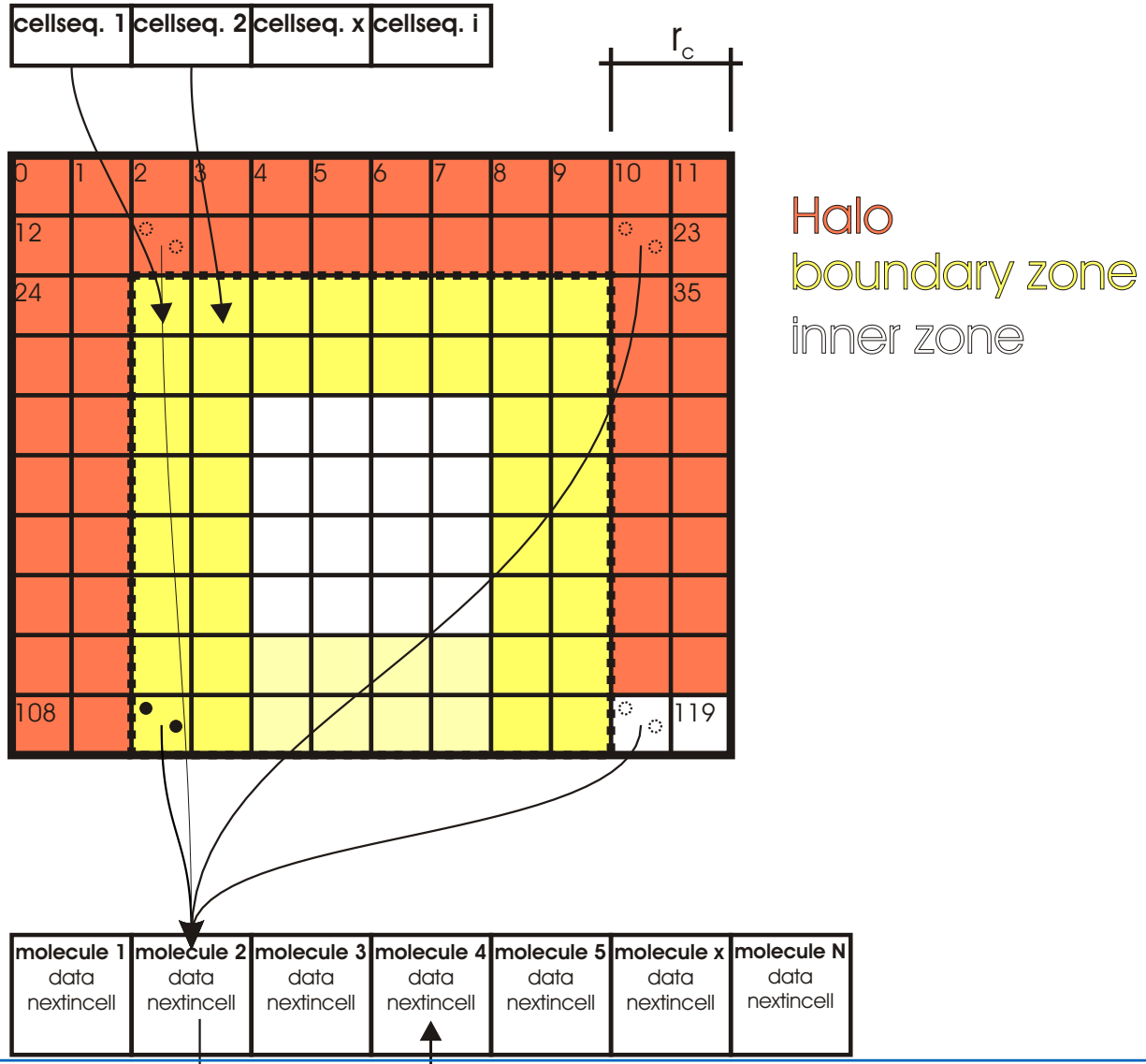
- molecules stored in a Cartesian grid (mesh size $\sim r_c$)
- only consider molecules in adjacent grid cells
- reduces computational effort to $\mathcal{O}(n)$
- only one half of the adjacent cells is considered
  $\rightarrow$ other forces are treated via symmetry of forces

Linked-Cell Algorithm (2)



- use geometric hash function to place molecules
- use “binning” and “bucketing” techniques from Computational Geometry
- important: update of data structure required after each time step!

Linked Cell – Data Structures

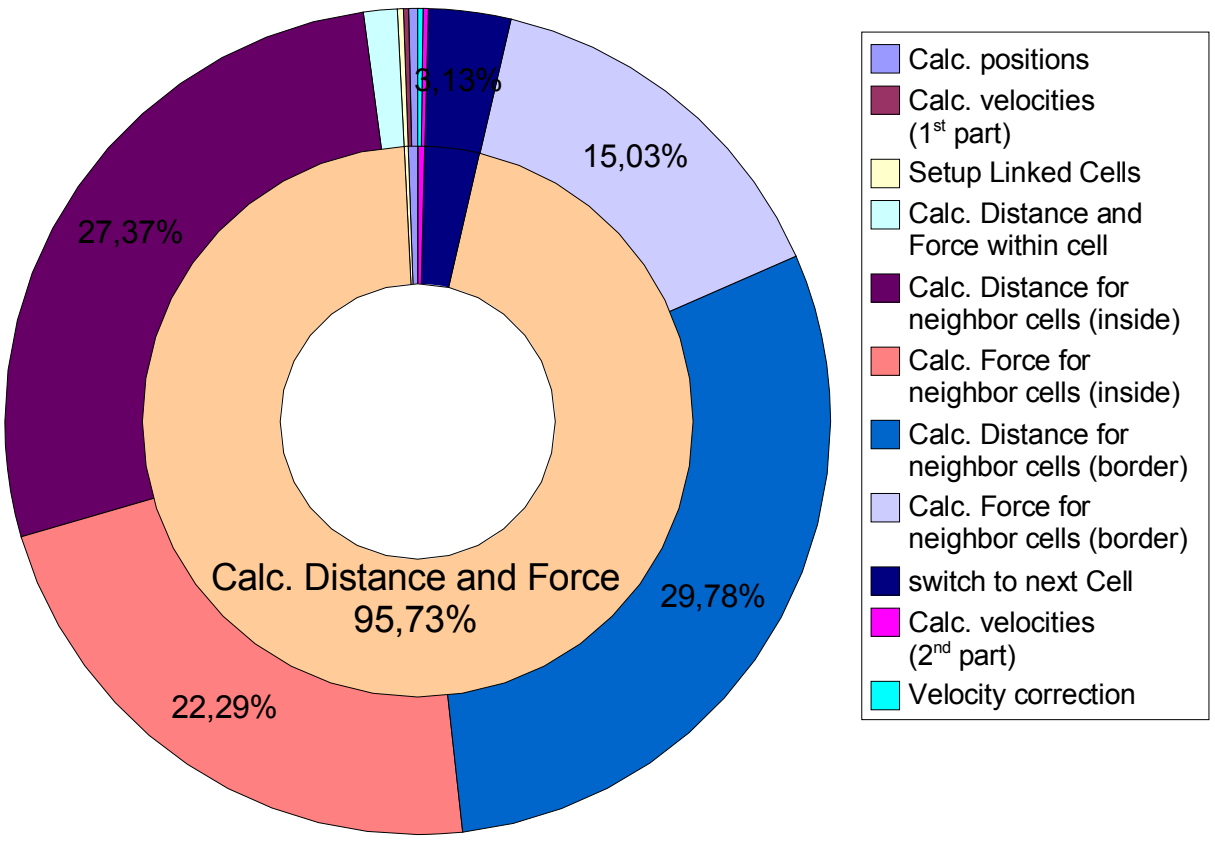


Linked Cell – Data Structures (2)

- cells stored as linearised array (1D/2D/3D)
- list implmentation for molecules within each cell
- “halo” and “boundary zone” (*ghost cells*) to treat periodic boundaries (replicate molecules)
- . . . or for parallelisation:
  “halo” replicates molecules of neighbouring partitions
  (requires communication between parallel processes)

Example of Execution Times

Profiling seq. MD code (LJ 12-6)



→ force computation dominates computational effort

Part II

Long-Range Forces

Long-Range Forces – Introduction

Long-Range Potentials

- Examples: Coulomb and gravity potential
- particles in large distance can contribute forces (if there are enough of them)
- formally: potential decaying slower than r^{-d} (criterion: d-dim. integration of the forces?)
- cut-off potential lead to large errors $\rightarrow$ how to avoid $\mathcal{O}(N^2)$ complexity?

Typical Case:

- mixture of short- and long-range forces
- represented as additive components:

$$U(r) := U^{\text{short}}(r) + U^{\text{long}}(r)$$

Mesh-Based Methods

Idea: separate U^{long} into two parts:

- smooth, long-range component
  $\rightarrow$ computes on a **mesh**
- non-smooth component with short range:
  $\rightarrow$ linked-cell approach (together with short-range forces)

Examples for mesh-based methods:

- P^3M (Particle-Particle–Particle-Mesh) method
- PME (Particle-Mesh-Ewald) method
- SPME (Smooth-Particle-Mesh-Ewald) method)

Mesh-Based Methods

PDE Formulation for potential Φ:

$$-\Delta\Phi(x) = \frac{1}{\epsilon_0}\varrho(x)$$

- potential computed from a potential equation (Poisson-type)
- solve with mesh-based discretisation
  $\rightarrow$ Finite Difference, Finite Elements, etc.
- useful for uniform distribution of particles

Hierarchical and Tree-based Methods

- starting point: integral representation of potential Φ

$$\Phi(x) = \frac{1}{4\pi\epsilon_0} \int \varrho(y) \frac{1}{\|y - x\|} dy$$

- hierarchical/adaptive algorithm for integration
- can be used for heterogeneous particle distribution (Astrophysics, also Molecular dynamics)
- Examples:
  - panel clustering
  - **Barnes-Hut**
  - (fast) multipole

Domain Decomposition

- distribute long-range region into subdomains:
  $\Omega^{\text{far}} = \bigcup_i \Omega_i^{\text{far}}$
- to be done for every particles position
  (in practice via hierarchical domain decomposition)
- assign a point y_0^i to each Ω_i^{far}
- decomposition depending on size of subdomains:

$$\text{diam} := \sup_{y \in \Omega_i^{\text{far}}} \|y - y_0^i\|$$

- choose decomposition such that

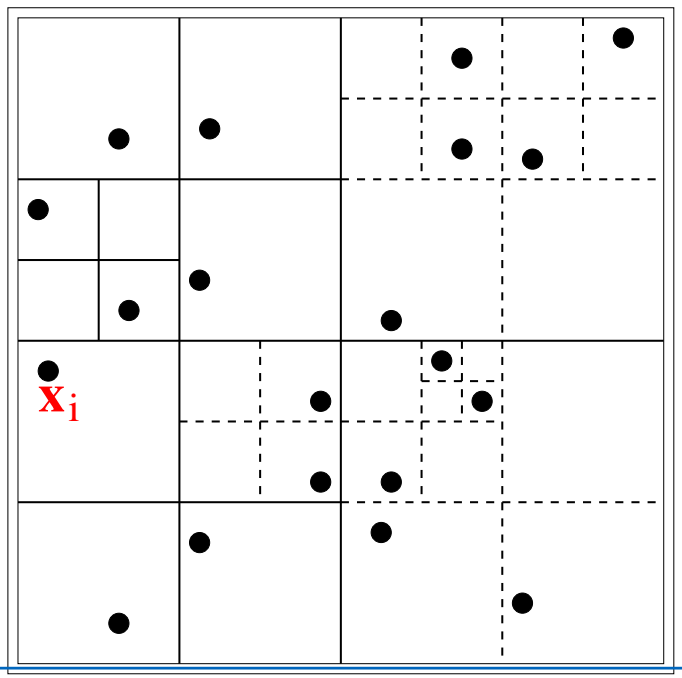
$$\frac{\text{diam}}{\|x - y_0^i\|} \leq \theta$$

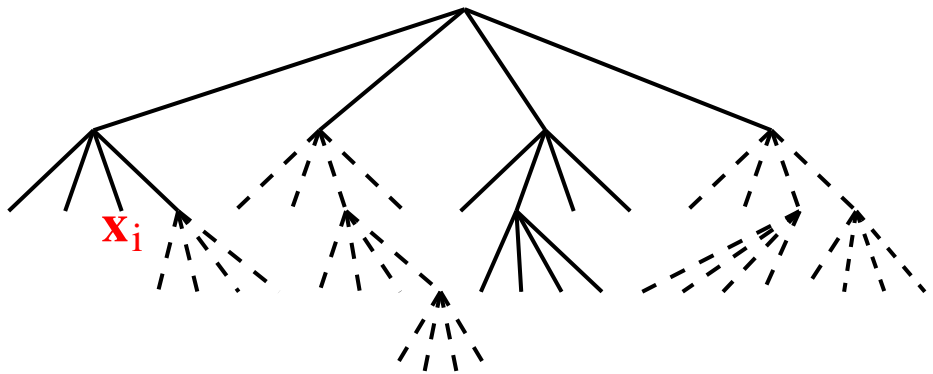
  for a suitable constant $0 < \theta < 1$

Octrees for Domain Decomposition

- efficient realisation of required decompositions
- recursive decomposition of Ω in subdomains
- stop, if only one particle left per cell
- use respective subtree for each x_i

Octrees:





Barnes-Hut Algorithm

- developed 1986 for applications in Astrophysics
- for gravity potential:

$$U(r_{ij}) = -\gamma_{\text{grav}} \frac{m_i m_j}{r_{ij}}$$

- uses octree with 0 or 1 particles per cell
- inner nodes corresp. to clusters of particles (**pseudo particle**)
- idea: gravity force of particle cluster approximated (sum of masses, localised in centre of mass)
- computation of forces: for each particle, do an incomplete(!) octree traversal

Barnes-Hut: Computation of Forces

For each particle (position $x \in \Omega$):

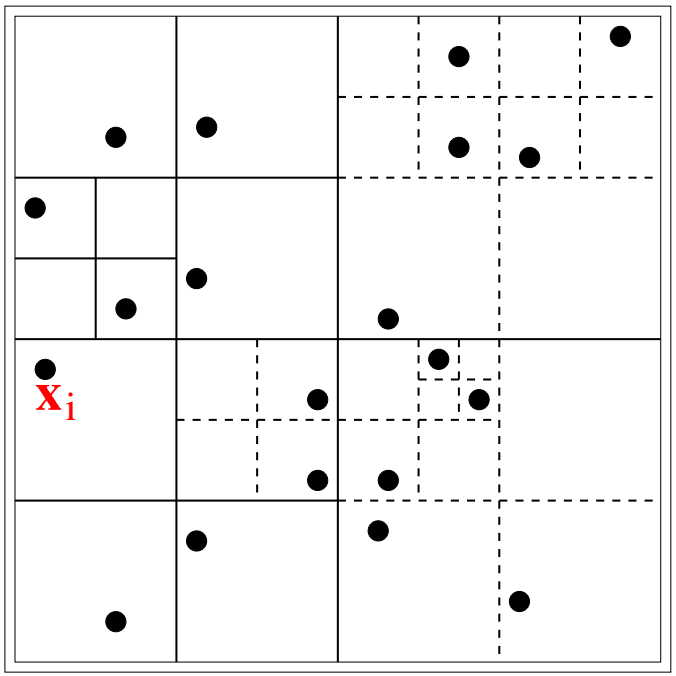
- start in root node
- decent into subdomains, until θ-rule satisfied: $\frac{diam}{r} \leq \theta$, r the distance of pseudo particle from x
- accumulate corresp. partial force to current particle

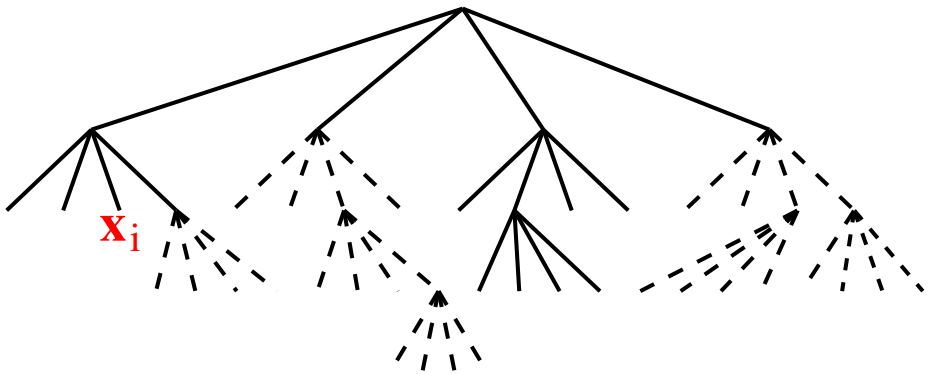
Implicit separation of short- and longe-range forces:

- short-range: all leaf nodes that are reached (containing 1 particle)
- long-range: all inner nodes, where decent is stopped (force caused by pseudo particle)

Barnes-Hut: Computation of Forces (2)

Tree traversal:





Barnes-Hut: Accuracy and Complexity

Accuracy of Barnes-Hut:

- depends on choice of θ
- the smaller θ, the more accurate the long-range forces
- the smaller θ, the larger the short-range (i.e., the costs)
- slow convergence w.r.t. θ (low-order method)

Complexity:

- grows for small θ
- for $\theta \to 0$: algorithm degenrates to "all-to-all" $\to \mathcal{O}(N^2)$
- for more or less homogeneously distributed particles:
  - number of active cells: $\mathcal{O}(\log N/\theta^3)$
  - total effort therefore $\mathcal{O}(\theta^{-3} N \log N)$

Barnes-Hut: Implementation

- computation of pseudo particles:
  - bottom-up-traversal (post-order)
  - sum up masses, weighted average for centre-of-mass
- computation of forces:
  - traversal of entire tree (outer loop on all particles)
  - top-down traversal (pre-order) until θ-rule satisfied (inner loop)
- further traversals for time integration
- re-build (or update) octree structure after each time step
  $\rightarrow$ requires efficient data structures and algorithms

Fast Multipole Methods

Barnes-Hut: forces of pseudo particles to particles
$\rightarrow$ Fast Multipole: forces between pseudo particles