KU LEUVEN

Fast Methods for Long-Range Many-Particle Interactions

G0L86A – Special Topics in Mathematics I

Paul Gibbon

KU Leuven and Forschungszentrum Jülich

2nd Semester, 11 February – 24 May, 2014

[]

0 – Contact details

Course ID:	G0B63A (Toledo)
email:	Paul.Gibbon@wis.kuleuven.be
tel:	016327002
room no.:	4.27
Skype:	paultgibbon
Office hours:	Wednesdays 14-16h (via Skype or Lync)

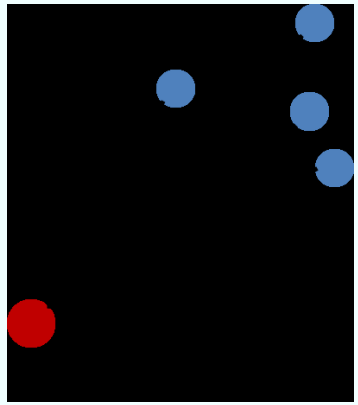
Part I

Introduction to long-range many-body problems

1. The N-body problem
2. Application areas
3. Numerical practice: simple N-body solver
4. Reading

1700s:	Planetary motion: General (exact) solution for $N > 2$ mutually gravitating bodies intractable
1870	Babbage builds 'Difference Engine' to compute astronomical tables for Nautical Almanac
1901	Kelvins assistent generates 5000 trajectories to compute elastic scattering of particles from curved walls
1908	W.S. Gossett estimates correlation coefficients with the help of ensemble averages
1945	von Neumann, Ulam find theoretical connection between probability and statistical ensemble averaging
1947	von Neumann, Ulam, Metropolis: use Monte-Carlo method to study diffusion of neutrons in fissile material
1953	Metropolis, Rosenbluth, Rosenbluth, Teller & Teller: Metropolis algorithm for idealised system (hard spheres or discs)

1957 Wood, Parker: MC for Lennard-Jones potentials
1959 Alder, Wainwright: Studies in Molecular Dynamics
1964 Aarseth: individual timesteps in gravitational N-body simulation
1978 Tremaine simulates interactions between galaxies
1980 Eastwood, Hockney, Lawrence: P^3M algorithm
1986 Barnes, Hut: Tree algorithm
1988 Greengard, Rohklin: Fast Multipole Method
1991 First parallel treecode (Salmon, Warren)



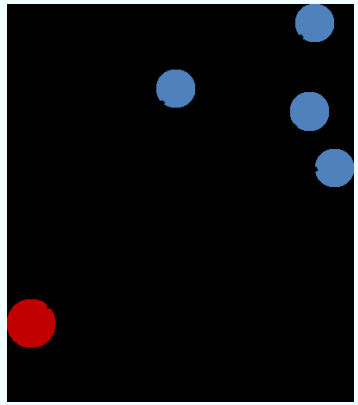
Consider interaction potential

$$V_{ij}(\boldsymbol{r}_i - \boldsymbol{r}_j) \equiv V(\boldsymbol{r}_{ij})$$

and let q_i be intrinsic particle property (usually charge or mass).

Then total potential energy of ensemble is:

$$U = \frac{1}{2}\sum_{i\neq j}^{N} q_i \sum_{j}^{N} V_{ij}, \qquad i,j = 1,2,3\ldots N. \tag{1}$$



The force acting on particle i is:

$$F(\boldsymbol{r}_i) = -q_i \nabla V(\boldsymbol{r}_i), \tag{2}$$

where

$$V(\boldsymbol{r}_i) = \sum_{j \neq i}^{N} V_{ij}.$$

Both equations (1) and (2) involve double summations over all N particles, omitting only a 'self interaction' term. Thus need to perform $N(N-1)$ operations, ie: consider $O(N^2)$ interactions for $N \gg 1$.

What does this mean in practice?

Example

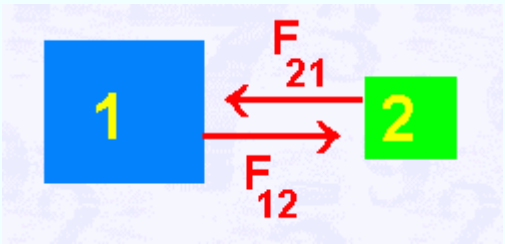
N	Time per force evaluation	10 000 steps
100	0.006 s	1 min
1 000	0.6 s	1.6 h
10 000	60 s	1 week
100 000	1.6 h	2 years

In N-body computations, statistical accuracy or signal/noise ratio often scales as $\sqrt{N}$, so direct or 'brute-force' technique simply not viable for problems where goal is to determine macroscopic properties of system.

Conservative systems – which can be described by a Hamiltonian – have the property:

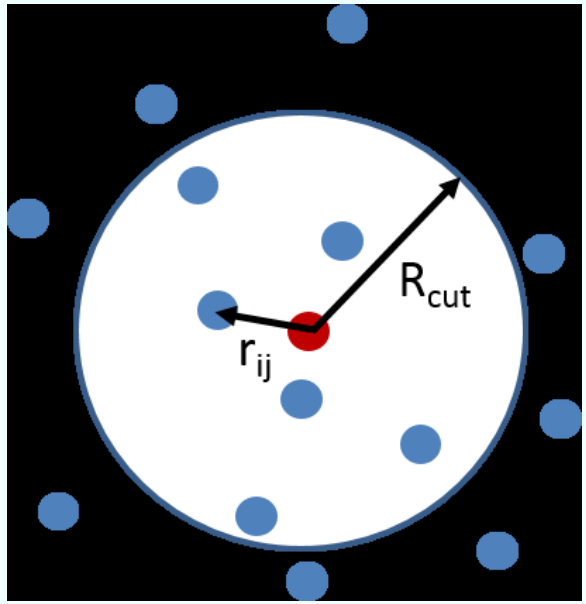
$$F_{ij} = -F_{ji} \tag{3}$$

– Newton's 3rd Law (actio=reactio)



Reduces sum in (2) by half; similar saving possible in (1), since $V_{ij} = V_{ji}$. Overall algorithm complexity is then $N(N-1)/2$.

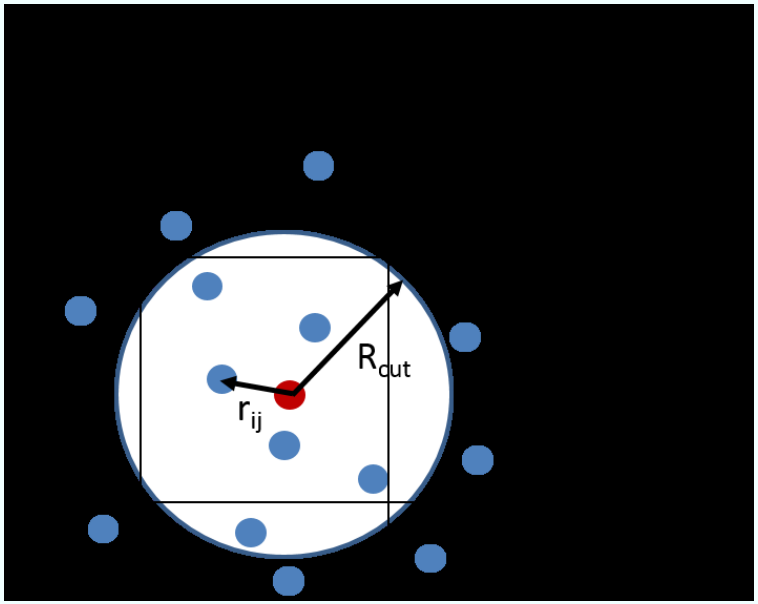
If potential or force-law is short-ranged, then a maximum interaction radius can be set.



Only interactions for which $|\boldsymbol{r}_{ij} < R_{\mathrm{cut}}|$ are included.

Determination of neighbour particles:

- Loop over all particles – $O(N^2)$!
- Verlet lists – $O(N^2)$, but updated after several steps/iterations
- Linked-cell lists $O(N)$



Will encounter linked-list scheme later in P^3M method.

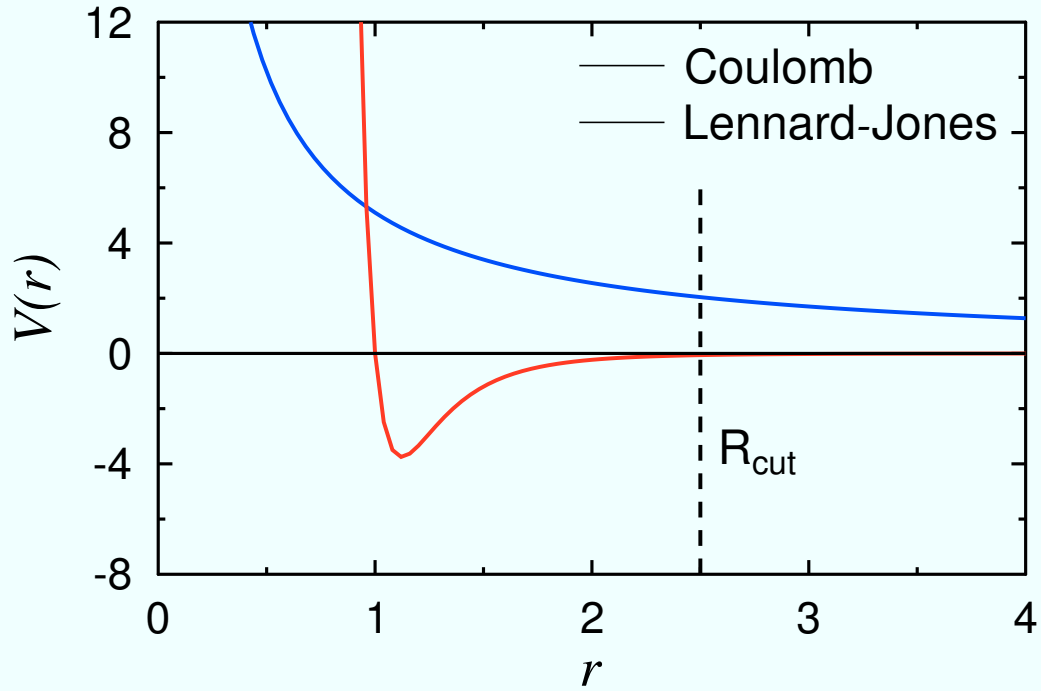
Truncation introduces error in force and potential: only practical where interaction is screened or has rapid fall-off.

Yukawa potential (cf: Debye screening in plasmas):

$$V_{ij} = \frac{a}{r_{ij}} \exp\left(-\alpha r_{ij}\right) \tag{4}$$

Lennard-Jones 12-6 potential:

$$V_{LJ} = \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \tag{5}$$



For Coulomb potential, cutoff will introduce discontinuity in net force felt by a particle – energy and momentum no longer conserved.

Exploiting computational power of modern supercomputers can in principle reduce time to solution of N^2 sum by sharing the problem among P processors.

$$T_{\text{solution}} \sim \frac{1}{2}\frac{N^2}{P}.$$

Buys time, but doesn't remove N^2 complexity!

Depending on physical context of problem, can replace N^2 sum with equivalent approximation with lower complexity:

Algorithm	Boundary conditions	Scaling
Ewald summation	periodic	$N^{3/2}$
Particle-particle, particle-mesh	periodic	N, $N \log N$
Tree method	open/periodic	$N \log N$
Fast multipole method	open/periodic	N

Approximation method is generally tunable, offering trade-off between speed and accuracy of force/potential sum.

Normally distinguish between absolute and relative errors.

- Absolute (average)

$$\epsilon_p = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(V_i^a - V_i^{\text{direct}} \right) \tag{6}$$

- Relative - force component F_k

$$\epsilon_k = \left\{ \frac{\sum_{i=1}^{N_{test}} \left(F_{ki}^a - F_{ki}^{\text{direct}} \right)^2}{\sum_{i=1}^{N_{test}} \left(F_{ki}^{\text{direct}} \right)^2} \right\}^{\frac{1}{2}} \tag{7}$$

where $N_{test} \leq N$, often $\ll N$ for very big systems.

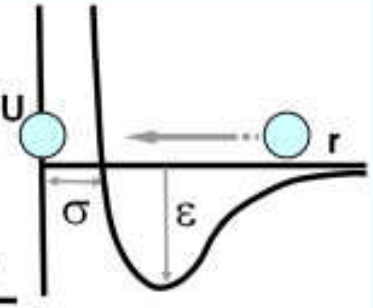
2 – Outline

2 – Where do N-body problems arise?

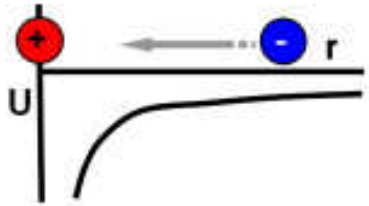
- *Ab initio* (from the beginning): 1 simulation particle = 1 physical
  - Classical molecular systems: proteins; chemical reactions
  - Bulk condensed matter: molten salts; ionic liquids, crystals
  - Particle beams (accelerators)
- *Continua* – described by particle methods through 'quasiparticles'
  - Plasmas
  - Gravitating systems (Newtonian)
  - Vortex fluids
  - Circuit design (potential maps)

2 – Example i) Molecular dynamics

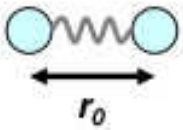
$$U = \sum_{i<j} \sum 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]$$



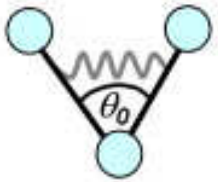
$$+ \sum_{i<j} \sum \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}$$



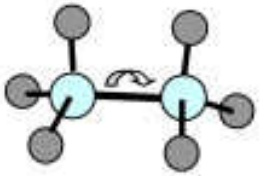
$$+ \sum_{bonds} \frac{1}{2} k_b (r - r_0)^2$$



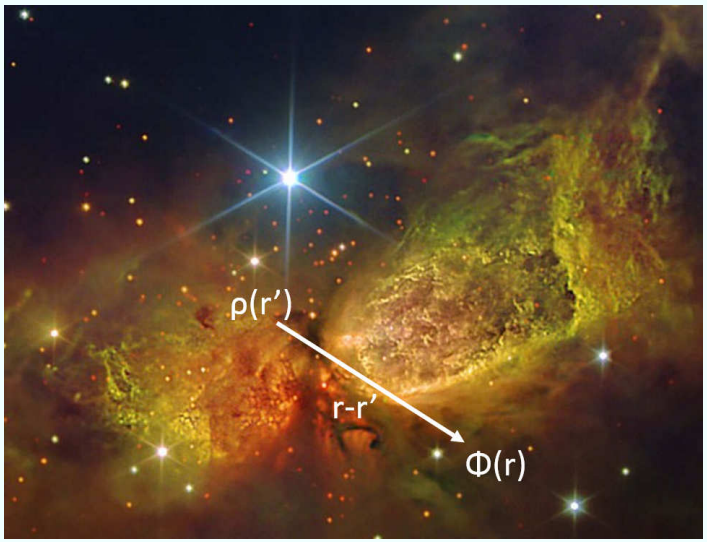
$$+ \sum_{angles} \frac{1}{2} k_a (\theta - \theta_0)^2$$



$$+ \sum_{torsions} k_\phi [1 + \cos(n\phi - \delta)]$$



Source: Scientific American (2012)



Potential due to arbitrary charge distribution given by Poisson's equation:

$$\nabla^2\phi(\boldsymbol{r}) \quad = \quad -4\pi G\rho(\boldsymbol{r}), \qquad r \in \mathbb{R}^3 \tag{8}$$

which has general solution

$$\phi(\boldsymbol{r}) \quad = \quad -\int_{V'} \frac{G\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} d^3r' \tag{9}$$

Can express integral (9) as discrete sum over mass elements $dm = \rho dV$, so that

$$\phi_i(\boldsymbol{r}_i, t) = -\sum_{j \neq i} \frac{G m_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \tag{10}$$

Corresponding dynamics is described by Newton's law:

$$\begin{aligned} m_i \frac{d\boldsymbol{v}_i}{dt} &= -m_i \nabla \phi(\boldsymbol{r}_i) = -G m_i \sum_{j \neq i} \frac{m_j (\boldsymbol{r}_i - \boldsymbol{r}_j)}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3} \\ \frac{d\boldsymbol{r}_i}{dt} &= \boldsymbol{v}_i. \end{aligned} \tag{11}$$

Need large number of particles to represenety density, fields and dynamics with reliable statistics. Resolution and/or size of simulation possible depends on how many particles a) can be stored, b) have interactions evaluated per step. State of the art: $N \sim 10^{12}$.

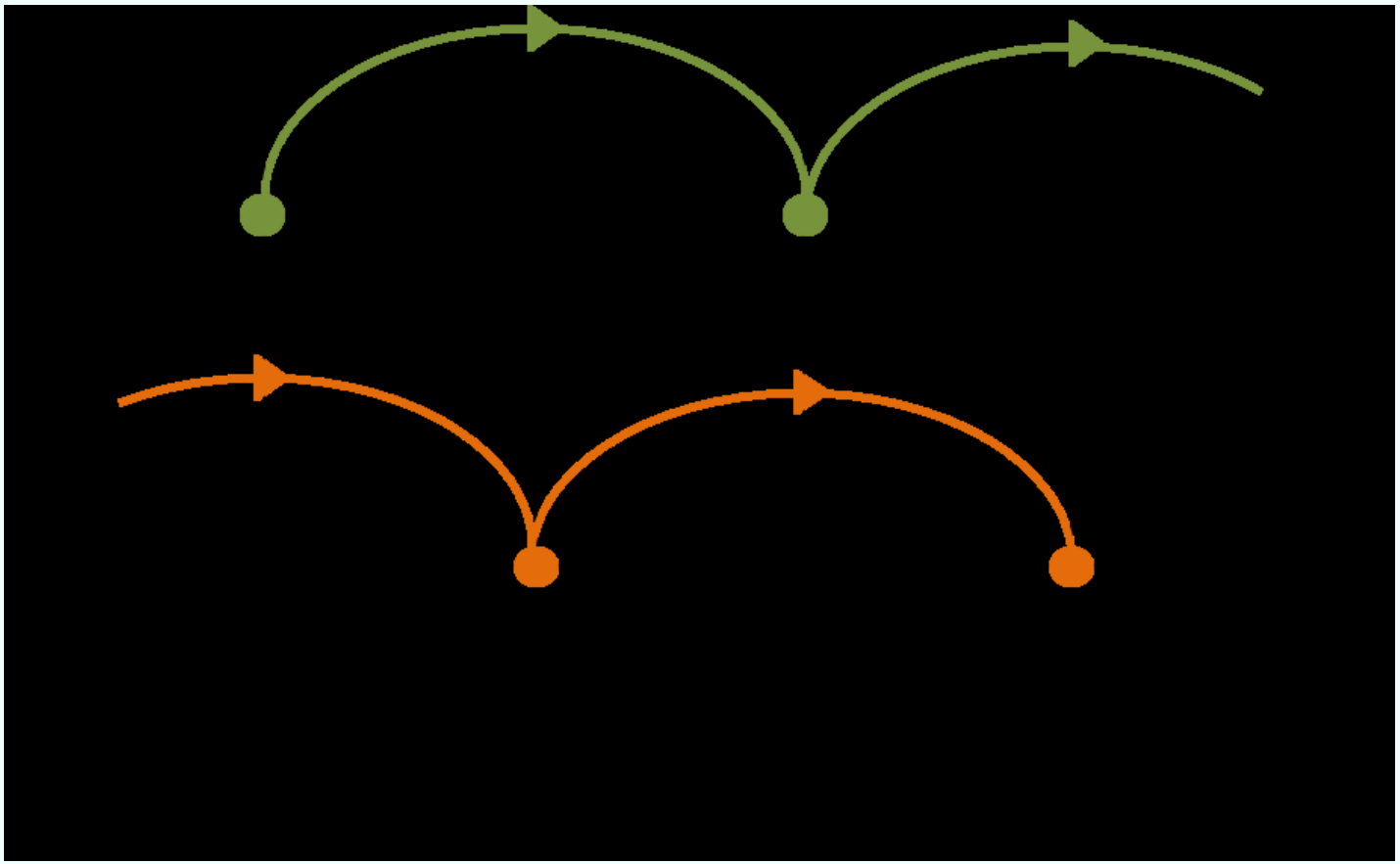
3 – Outline

To solve the dynamical system described by Eq. (12), we can apply a simple Leap-Frog scheme – one of many possible finite-difference integrators.
For simplicity, we restrict the motion to a the (x, y) plane.
Our velocity components can thus be computed as follows:

$$\begin{aligned} v_x^{n+\frac{1}{2}} &= v_x^{n-\frac{1}{2}} - G\Delta t \sum_{j\neq i} \frac{m_j(x_i^n - x_j^n)}{(|\boldsymbol{r}_i^n - \boldsymbol{r}_j^n|^2 + \epsilon^2)^{3/2}} \\ v_y^{n+\frac{1}{2}} &= v_y^{n-\frac{1}{2}} - G\Delta t \sum_{j\neq i} \frac{m_j(y_i^n - y_j^n)}{(|\boldsymbol{r}_i^n - \boldsymbol{r}_j^n|^2 + \epsilon^2)^{3/2}} \end{aligned} \tag{12}$$

To complete the integration, we update the particle positions with the freshly computed velocities:

$$\begin{aligned} x^{n+1} &= x^n + v_x^{n+\frac{1}{2}} \Delta t \\ y^{n+1} &= y^n + v_y^{n+\frac{1}{2}} \Delta t \end{aligned} \tag{13}$$



To avoid singular field/potential values, potential is softened by a smoothing parameter ϵ. System is thus governed by 'Plummer', not Coulomb force law: approximation, but still long-ranged. Normally, would have to include 'hard-sphere' limit for some minimum $\Delta \boldsymbol{r}$, then adjust timestep on the fly according to trajectory.
For simplicity, consider quasi-collisionless system here, such that:

$$\begin{aligned} \dot{v}_{max} &\simeq \frac{\Delta v}{\Delta t} = \frac{Gm}{2\epsilon^2} \\ \Delta t_{max} &= (\Delta v)_{max} \cdot \frac{2\epsilon^2}{Gm} \end{aligned} \tag{14}$$

A simple N-body solver, ngrav.py is provided to get you started with the assignment. The code sets up a randomly distributed disc of particles, and computes their forces according to (12). This is done via a straightforward double-loop over the particle indices:

ngrav.py

```
do i=1,N
   do j=1,N
      if (j<>i)
         sum pair force
   end do
   update velocities
   update positions
end do
```

- Using the ngrav.py Python script provided or your own program in a high-level language of your choice, modify the main force-summation loop to compute the potential energy U_{pot} (according to Eq.10) of the system at each timestep.
- Now add the kinetic energy

$$U_{kin} = \frac{1}{2} \sum_i m_i v_i^2,$$

and the total energy

$$U_{tot} = U_{pot} + U_{kin}$$

- Run your N-body disc setup for a number of timesteps (eg 100) and check the energy conservation.
- Measure the time taken per run for the force summation loop and vary the number of particles (careful not to increase N too fast!) to check how the computational effort scales with N.
- Now try and optimize your algorithm: are there any redundant floating-point operations? Apply the force symmetry Eq. (3) to halve the number of force/potential computations. Check that your new code i) still conserves energy, ii) is truly 2x faster!

4 – Outline

- G. Sutmann, P. Gibbon, Th. Lippert (eds): *Fast Methods for Long-Range Interactions in Complex Systems*, IAS Series, Forschungszentrum Jülich, ISSN 1868-8489
- R. W. Hockney & J. W. Eastwood, *Computer Simulation using Particles*, Taylor & Francis, New York (1988).
- S. Pfalzner & P. Gibbon, *Many Body Tree Methods in Physics*, Cambridge University Press (1996, 2005) ISBN 0-521-01916-8
- J. Barnes & P. Hut, *A hierarchical O(N log N) force-calculation algorithm*
- L. Greengard & Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys. **73**, 325 (1987).

Part II

Ewald Summation

5 Periodic systems

6 Minimum image

7 Ewald summation

8 Optimisation

5 – Outline

- Techniques which rely on the fact that a large class of physical problems can be simplified by periodic boundaries to eliminate artificial surface effects for very large systems (bulk solids, liquids).
- Reduces number of particles needed to yield statistically meaningful predictions of physical/material characteristics – eg: structural properties, equation of state, electrical/thermal conductivity.
- Idea is to exploit translational sysmmetry $f(x \pm nL) = f(x)$, allowing Fourier transforms to be applied to handle 'far field' of interaction region

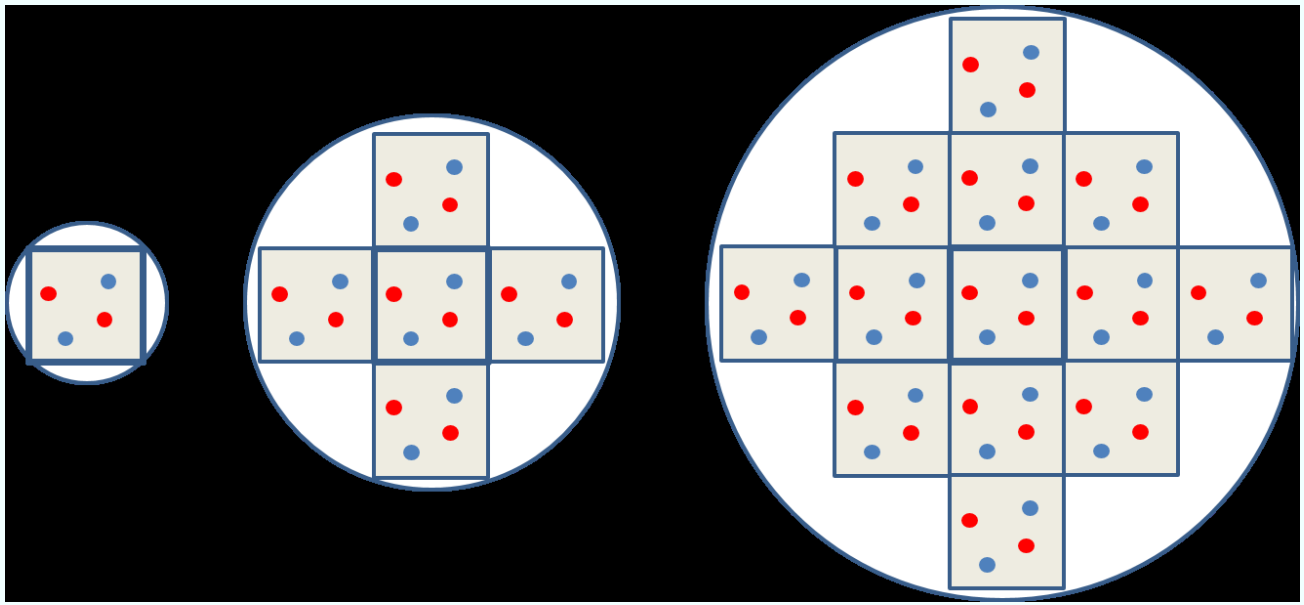
Periodic continuation of physical region of interest immediately poses question of summation ordering. Suppose our system of particles is constrained within a cubic box of length L.
According to this scheme, the potential is then expressed mathematically as:

$$V_s(r_i) = \sum_{\boldsymbol{n}} {}' \sum_{j=1}^{N} \frac{q_j}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid}, \tag{15}$$

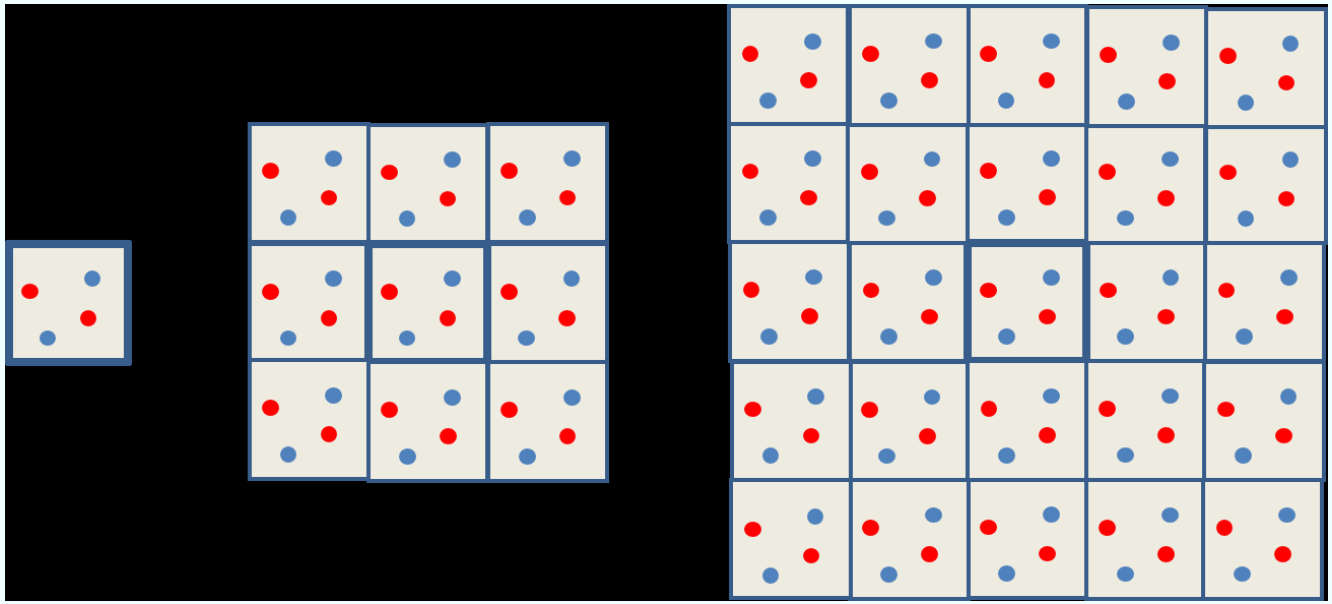
where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and $\boldsymbol{n} = (i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})L$, with $i_\alpha = 0, \pm 1, \pm 2 ... \pm \infty$. The prime in the sum over $\boldsymbol{n}$ indicates that the $j = i$ term is omitted for the primary cell $\boldsymbol{n} = 0$, but otherwise included for the image cells.

This is a conditionally convergent series: depends on the order of summation!
NB: cf. potentials with $r^{-\alpha}, \alpha > 3$, which are absolutely convergent.

One choice of ordering is to build up sets of images contained within successively larger spheres surrounding the simulation region:

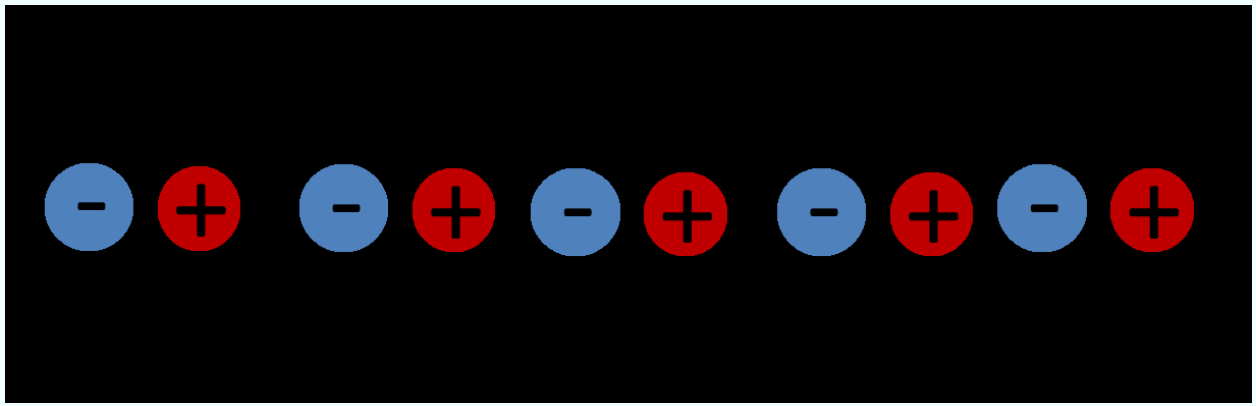


Another choice would be to retain the geometry of the primary cell and use nested cubes:



The problem: these two orderings will give different results for the total potential energy!

A simple example serves to illustrate this problem. Consider a system of two oppositely charged ions, periodically extended to form an infinite one-dimensional line of charges, each separated by a distance R:



The potential energy of the reference ion with charge $-q$ is:

$$\begin{aligned} U &= -2q^2\left(\frac{1}{R}-\frac{1}{2R}+\frac{1}{3R}-\frac{1}{4R}\ldots\right) \\ &= -\frac{2q^2}{R}\left(1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}\ldots\right) \\ &= -\frac{2q^2}{R}\log 2 \end{aligned} \tag{16}$$

The factor $2\log 2$ is the Madelung constant, which is of central importance in the theory of ionic crystals.

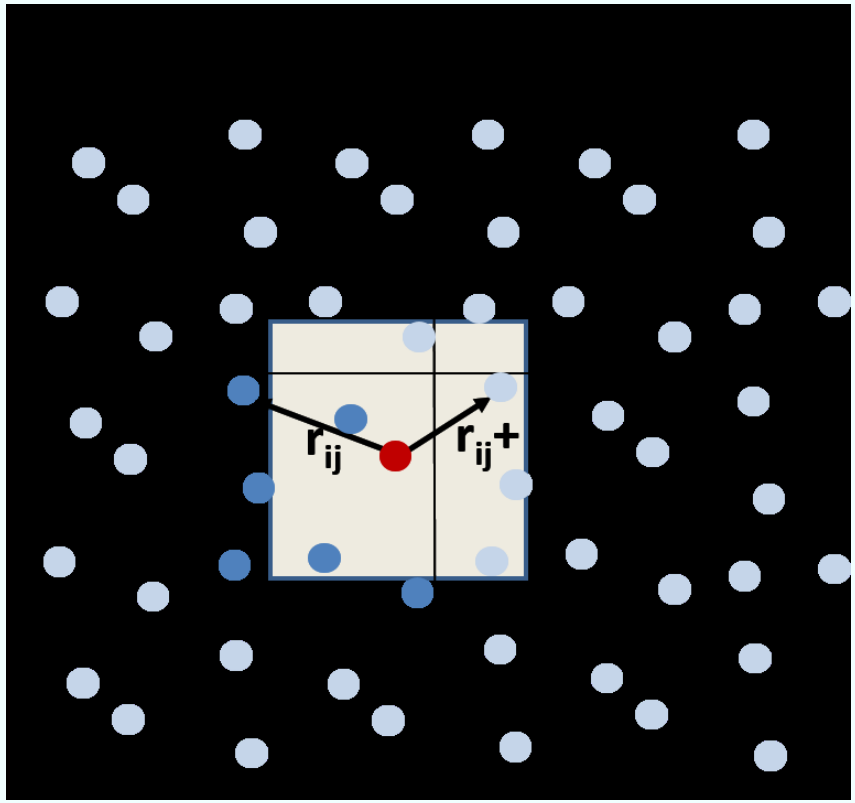
The series in (16) is also *conditionally convergent*; the result depends on the summation order. To illustrate this, we can choose a different ordering, for example:

$$
\begin{aligned}
& 1 \quad + \frac{1}{3} - \frac{1}{2} + \frac{1}{5} \quad + \frac{1}{7} - \frac{1}{4} + \frac{1}{9} \quad + \frac{1}{11} - \frac{1}{6} + \frac{1}{13} \cdots \\
= \; & 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \frac{1}{9} - \frac{1}{10} \cdots \\
& + \frac{1}{2} \quad - \frac{1}{4} \quad + \frac{1}{6} \quad - \frac{1}{8} \quad + \frac{1}{10} \cdots \\
= \; & \sum_n \frac{(-1)^{n-1}}{n} + \frac{1}{2} \sum_n \frac{(-1)^{n-1}}{n} = \frac{3}{2} \log 2,
\end{aligned}
$$

giving us 50% more potential energy than we had before!

6 – Outline

What if we just restrict the periodic extension to a single box but shift origin?



Images further away from particle than box length are ignored – ie: for separation take:

$$d_{ij} = \min\{|\,\boldsymbol{r}_{ij}\,|, |\,\boldsymbol{r}_{ij} + \boldsymbol{n}\,|, |\,\boldsymbol{r}_{ij} - \boldsymbol{n}\,|\} \tag{17}$$

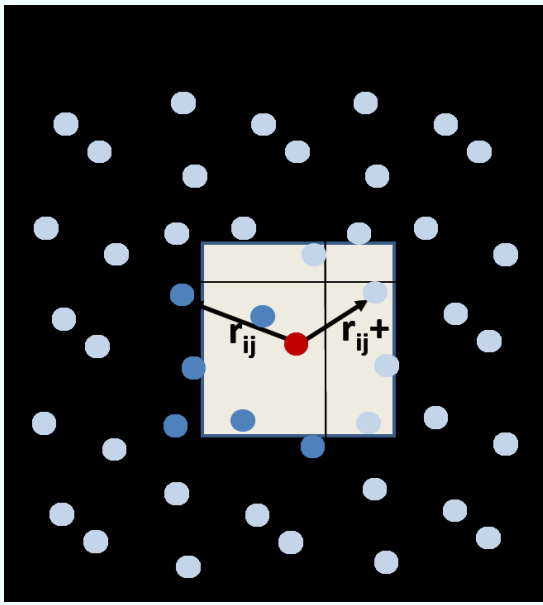
System is internally consistent with $\boldsymbol{F} = -q\nabla\Phi$, but get different potential energy and forces from fully periodic system (infinite no. images) described by Eq. 15 – see **Assignment A2**.

To obtain high precision in both potential and forces, the minimum image convention won't suffice, and we need to include a large number of image boxes. On the face of it, therefore, things have gone from bad to worse: we have turned an $O(N^2)$ algorithm into $O(N_{box}N^2)$!

6 – Assignment A2: Periodic boundaries using ‘minimum-image’ truncation

- Modify your (by now highly tuned) ngrav.py script to describe a charged particle system instead of masses. Set up the charges randomly in a square box such that $\sum q = 0$. The relevant equations are:

$$\Phi = \frac{q}{r} \quad F = -q\nabla\phi, \quad \frac{d\boldsymbol{v}}{dt} = \frac{\boldsymbol{F}}{m}; \quad \frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

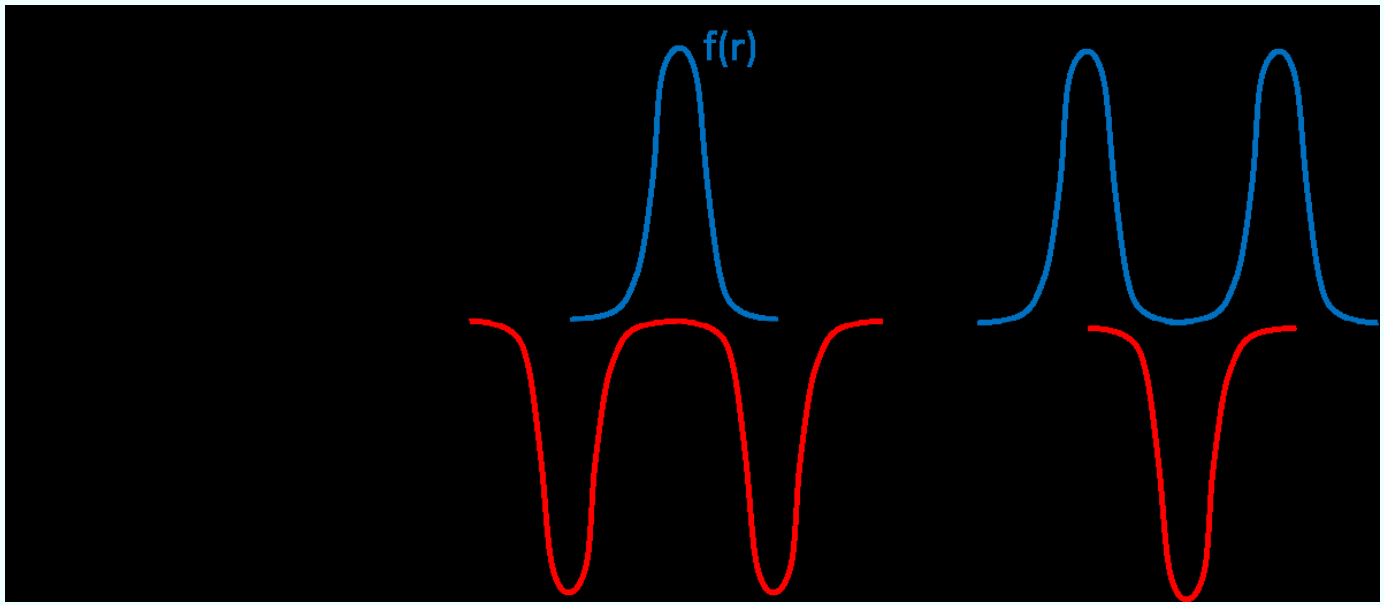


- Introduce periodic boundaries into the particle dynamics This is best done after the position update, modifying each coordinate so that particles leaving the box are ‘wrapped’ onto the opposite side. Run this system with a few particles (10-20) and check the energy conservation: comment on the behaviour.
- Now make the *forces* periodic via the minimum image scheme, modifying the summation region by including particles $\leq \pm L/2$ distant from the particle of interest and recheck the energy conservation.
- Test the convergence of the force sum for the 1st iteration (set nstep=1) by including additional image boxes at a) 2L and b) 3L. (Use just a few particles for this).

7 – Outline

P.P. Ewald (1921) pointed out that the slowly converging series of (15) could be rewritten as sum of two rapidly converging sums. Using a simple splitting scheme:

$$\frac{1}{r} = \frac{f}{r} + \frac{1-f}{r}, \tag{18}$$



Adding and subtracting additional series with finite form factor (kernel) $f(r)$, the potential summation can be rewritten:

$$V_E(r_i) = V_s(r_i) - \sum_{\boldsymbol{n}} f(\boldsymbol{n}) + \sum_{\boldsymbol{k}} g(\boldsymbol{k}). \tag{19}$$

where $f(\boldsymbol{n})$ and $g(\boldsymbol{k})$ are summed in real space and k-space respectively. Ewald originally used a Gaussian kernel:

$$\sigma(r) = \frac{\alpha^3}{\pi^{3/2}} \exp(-\alpha^2 r^2), \tag{20}$$

which satisfies

$$\int_0^\infty \sigma(r) dr = 1,$$

and α determines the height/width ratio of the spreading function (effective size of charges).

Using the Gaussian kernel, the potential summation (15) becomes:

$$
\begin{aligned}
V_E(r_i) &= V_E^r + V_E^k + V_E^s \\
&= \sum_{\boldsymbol{n}}{}' \sum_{j=1}^{N} q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \\
&+ \frac{4\pi}{L^3} \sum_{\boldsymbol{k} \neq 0} \sum_{j} q_j \frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{\mid \boldsymbol{k} \mid^2} \exp\left\{i\boldsymbol{k} \cdot (\boldsymbol{r}_j - \boldsymbol{r}_i)\right\} \qquad (21) \\
&- \frac{2\alpha}{\pi^{1/2}} q_i
\end{aligned}
$$

where V_E^s compensates for a self-term in the reciprocal-space part V_E^k.

Real space: first subtract lattice sum for smeared-out charges with kernel $\sigma(r)$ from original point-charge sum:

$$
\begin{aligned}
V_E^r(r_i) &= \sum_{\boldsymbol{n}}{}' \sum_{j=1}^{N} \frac{q_j}{r_{ij,\boldsymbol{n}}} \left[1 - \int_0^\infty \sigma(r - r_{ij}) d^3 r \right] \\
&= \sum_{\boldsymbol{n}}{}' \sum_{j} q_j \left[\frac{1}{r_{ij,\boldsymbol{n}}} - \frac{4\alpha^3}{\pi^{1/2} r_{ij,\boldsymbol{n}}} \int_0^{r_{ij,\boldsymbol{n}}} r^2 \exp(-\alpha^2 r^2) dr \right. \\
&\qquad \left. - \frac{4\alpha^3}{\pi^{1/2}} \int_{r_{ij,\boldsymbol{n}}}^\infty r \exp(-\alpha^2 r^2) dr \right] \qquad (22)
\end{aligned}
$$

where we have used the shorthand notation $r_{ij,\boldsymbol{n}} \equiv | \boldsymbol{r}_{ij} + \boldsymbol{n} |$

Exercise

The *2nd term* in (22) can be integrated by parts to get to give an error function plus a term which exactly cancels the third term, finally yielding for the real-space part:

$$\begin{aligned} V_E^r &= \sum_{\boldsymbol{n}}{}' \sum_{j=1} q_j \left[\frac{1}{r_{ij,\boldsymbol{n}}} - \frac{\mathrm{erf}(\alpha r_{ij,\boldsymbol{n}})}{r_{ij,\boldsymbol{n}}} \right] \\ &= \sum_{\boldsymbol{n}}{}' \sum_{j=1} q_j \frac{\mathrm{erfc}(\alpha r_{ij,\boldsymbol{n}})}{r_{ij,\boldsymbol{n}}} \end{aligned} \tag{23}$$

Reciprocal space: Consider charge density of whole lattice at position r:

$$\rho(r) = \sum_j q_j \delta(r - r_j). \tag{24}$$

Since the lattice is periodic, we can express this equivalently as a Fourier sum:

$$\rho(r) = L^{-3} \sum_{\boldsymbol{k}} f(\boldsymbol{k}) \exp(-i\boldsymbol{k} \cdot \boldsymbol{r}), \tag{25}$$

where $\boldsymbol{k} = 2\pi/L(i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})$; $i_\alpha = 0, \pm 1, \pm 2,$ ..etc., and

$$f(\boldsymbol{k}) = \int_{L^3} \rho(r) \exp(i\boldsymbol{k} \cdot \boldsymbol{r}) d^3 r, \tag{26}$$

Substituting (54) into (26) gives

$$\begin{aligned} f(\boldsymbol{k}) &= \int_{L^3} \sum_j q_j \delta(r - r_j) \exp(i\boldsymbol{k} \cdot \boldsymbol{r}) d^3 r \\ &= \sum_j q_j \exp(i\boldsymbol{k} \cdot \boldsymbol{r}_j) \end{aligned} \tag{27}$$

Turning now to the *smeared* charge distribution:

$$\begin{aligned} \rho'(r) &= \sum_j q_j \sigma(r - r_j) \\ &= \int_V \rho(r - r') \sigma(r') d^3 r', \end{aligned} \tag{28}$$

Exercise

Eq. (28) is just the convolution of $\rho(r)$ with $\sigma(r)$, which we know can be expressed in Fourier space as:

$$\rho'(r) = \frac{1}{L^3} \sum_{\boldsymbol{k}}{}' f(\boldsymbol{k}) \phi(\boldsymbol{k}, \alpha) \exp(-i\boldsymbol{k} \cdot \boldsymbol{r}), \tag{29}$$

where $\phi(\boldsymbol{k}, \alpha)$ is the Fourier transform of the charge-smearing function $\sigma(r)$ from (20), i.e.:

$$\phi(\boldsymbol{k}, \alpha) = \exp\left(\frac{-\mid \boldsymbol{k} \mid^2}{4\alpha^2}\right). \tag{30}$$

Note: prime in $\sum_{\boldsymbol{k}}{}'$ indicates that $\boldsymbol{k} = 0$ term omitted.

Potential due to the smeared charges in k-space can now be evaluated. At the reference position r_i, this is:

$$\begin{aligned} V_k(r_i) &= \int_0^\infty \frac{\rho'(r_i + r)}{r} d^3r \\ &= \frac{1}{L^3} {\sum_{\boldsymbol{k}}}' f(\boldsymbol{k}) \phi(\boldsymbol{k}, \alpha) \exp(-i\boldsymbol{k} \cdot \boldsymbol{r}_i) \int_0^\infty \frac{\exp(-i\boldsymbol{k} \cdot \boldsymbol{r})}{r} d^3r. \end{aligned} \tag{31}$$

The integral on the right of the above expression is a standard one and evaluates to $4\pi/k^2$.

Combining (31) with the earlier results (27) and (30) for $f(\boldsymbol{k})$ and $\phi(\boldsymbol{k}, \alpha)$ respectively, we have finally:

$$V_E^k(r_i) = \frac{4\pi}{L^3} \sum_{\boldsymbol{k}}{}' \sum_j q_j \exp\{i\boldsymbol{k} \cdot (\boldsymbol{r}_j - \boldsymbol{r}_i)\} \frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{|\ \boldsymbol{k}\ |^2}. \qquad (32)$$

One final correction is necessary to (32), because it contains an unphysical 'self-potential' of smeared charge at $\boldsymbol{r}_i$, which has to be subtracted off:

$$\begin{aligned} V_s(r_i) &= q_i \int_0^\infty \sigma(r) d^3 r \\ &= \frac{4\pi q_i \alpha^3}{\pi^{3/2}} \int_0^\infty r \exp(-\alpha^2 r^2) \\ &= \frac{2\alpha}{\pi^{1/2}} q_i. \end{aligned} \tag{33}$$

Combining partial results (23, 32, 33) recovers the initial expression (21)

Exercise

The corresponding electric field components can be found simply by differentiating Eq.(21) with respect to the vector between the reference particle i and particle j:

$$\begin{aligned}
\boldsymbol{E}(\boldsymbol{r}_i) &= -\frac{\partial V_E(\boldsymbol{r}_i)}{\partial \boldsymbol{r}_{ij}} \\
&= \sum_{\boldsymbol{n}}{}' \sum_j \frac{q_j \boldsymbol{r}_{ij,\boldsymbol{n}}}{r^3_{ij,\boldsymbol{n}}} \left[\mathrm{erfc}(\alpha r_{ij,\boldsymbol{n}}) + \frac{2\alpha r_{ij,\boldsymbol{n}}}{\sqrt{\pi}} \exp(-\alpha^2 r^2_{ij,\boldsymbol{n}}) \right] \\
&+ \frac{4\pi}{L^3} \sum_{\mathbf{k}\neq 0} \sum_j q_j \frac{\boldsymbol{k}}{k^2} \exp\left(\frac{-k^2}{4\alpha^2}\right) \sin\left(\boldsymbol{k}\cdot\boldsymbol{r}_{ji}\right). \qquad (34)
\end{aligned}$$

Shorthand notation: $\boldsymbol{r}_{ij,\boldsymbol{n}} \equiv \boldsymbol{r}_{ij} + \boldsymbol{n}$, $r_{ij,\boldsymbol{n}} = \mid \boldsymbol{r}_{ij,\boldsymbol{n}} \mid$, $\boldsymbol{r}_{ji} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$

8 – Outline

Both summations in (21) converge unconditionally and rapidly, as long as we choose an appropriate value of the dimensionless parameter αL.
Thus, in practice we can truncate both sums at some multiple of the box length $|\boldsymbol{n}|_{max}$ and restrict the wave vectors to values below a given k_{max}:

$$\begin{aligned} V_E(r_i) &= V_E^r + V_E^k + V_E^s \\ &= \sum_{\boldsymbol{n}}^{\boldsymbol{n}_{max}} {}' \sum_{j=1}^{N} q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \\ &+ \frac{4\pi}{L^3} \sum_{\boldsymbol{k}\neq 0}^{k_{max}} \sum_{j} q_j \frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{\mid \boldsymbol{k} \mid^2} \exp\left\{i\boldsymbol{k}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)\right\} \qquad (35) \\ &- \frac{2\alpha}{\pi^{1/2}} q_i \end{aligned}$$

First re-introduce integer-triple from p. 50:

$$\boldsymbol{h} \equiv \frac{L}{2\pi}\boldsymbol{k}$$

where $\boldsymbol{h} = (i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})$; $\quad i_\alpha = 0, \pm 1, \pm 2,$ ..etc. Then let

$$A(\boldsymbol{h}) = \frac{4\pi^2}{L^2 \mid \boldsymbol{h} \mid^2} \exp\left(\frac{-\pi^2 \mid \boldsymbol{h} \mid^2}{\alpha^2 L^2}\right) \tag{36}$$

Total potential energy of system is

$$U = \frac{1}{2}\sum_{i}^{N} q_i V_E(r_i) = U_r + U_k - U_s$$

with

$$\begin{aligned}
U_r &= \frac{1}{2}\sum_{\boldsymbol{n}}^{\boldsymbol{n}_{max}}{}' \sum_i \sum_j q_i q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \\
U_k &= \frac{1}{2\pi L}\sum_{\boldsymbol{h}\neq 0} A(\boldsymbol{h}) \left\{ \left(\sum_i q_i \cos \frac{2\pi}{L} \boldsymbol{h} \cdot \boldsymbol{r}_i \right)^2 + \left(\sum_i q_i \sin \frac{2\pi}{L} \boldsymbol{h} \cdot \boldsymbol{r}_i \right)^2 \right\} \\
U_s &= -\frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2
\end{aligned} \tag{37}$$

Exercise

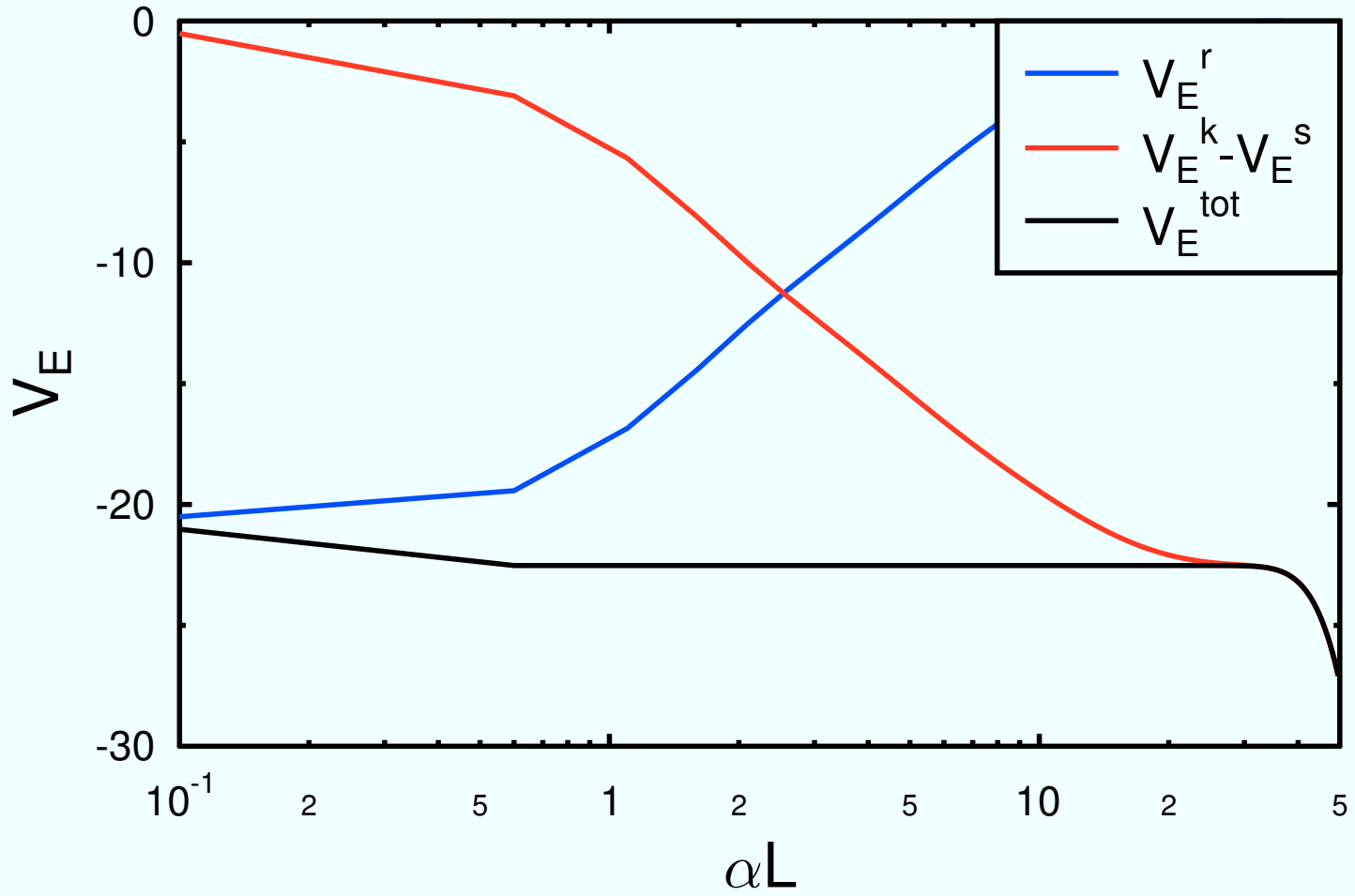
The k-space part of the potential has been reduced to a single sum over the particles. Use the identity

$$\cos(A - B) = \cos A \cos B + \sin A \sin B$$

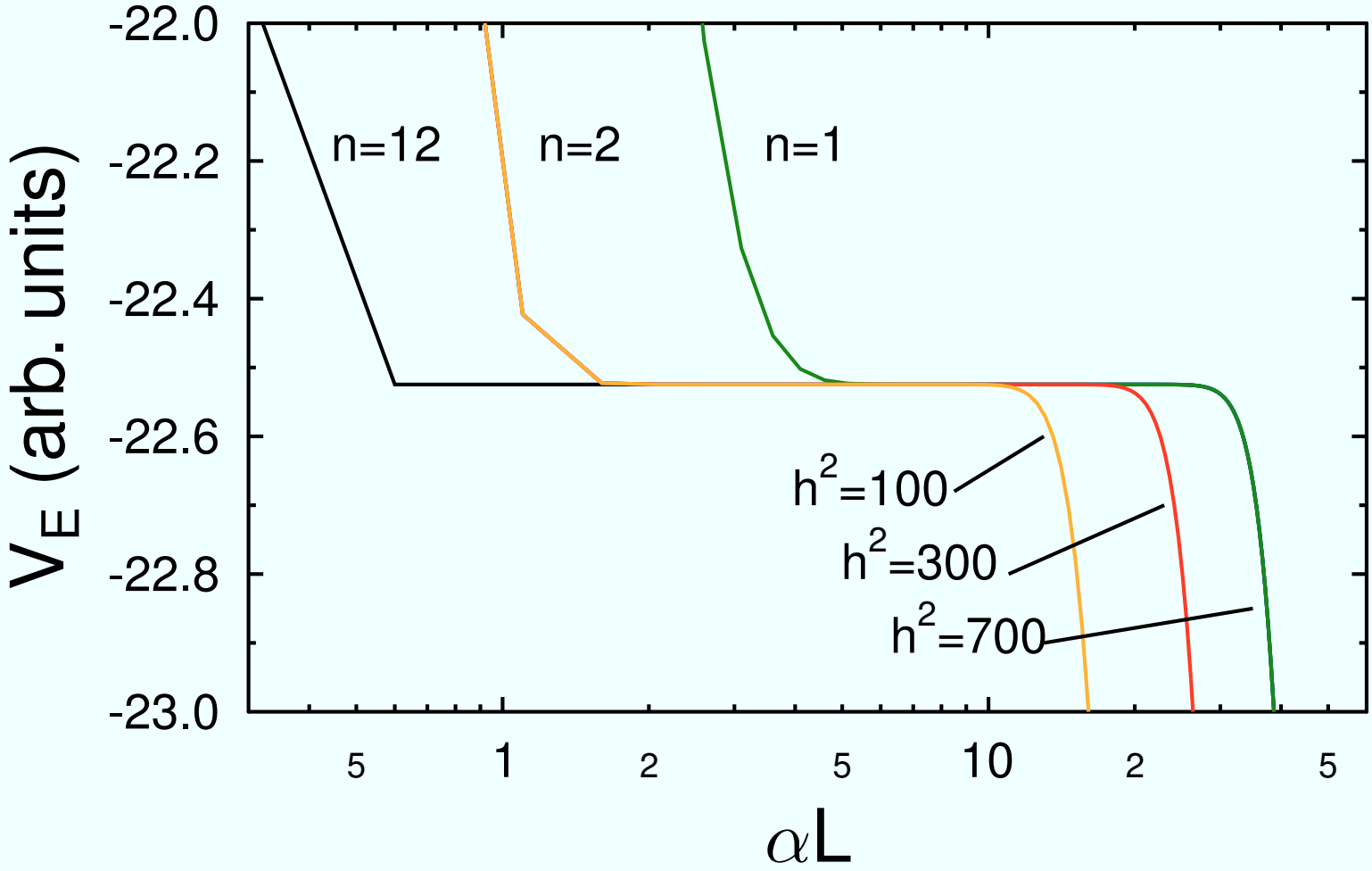
and reorder the particle summations to show this explicitly.

8 – Summation example

Example: 40 randomly distributed charges (+ve and -ve), with $|\boldsymbol{n}|_{max} = 12, h^2 = (k_{max}/2\pi)^2 = 700$, respectively.



Same charge distribution as before, now with differing truncations in the real- and k-space sums.



As we have seen, the relative weight of the computational effort in the real- and k-space sums depends on the choice of αL. At one extreme, minimum image, where $\mid \boldsymbol{r}_i + \boldsymbol{n} \mid \leq L$, have effort $O(N^2)$.
This can be reduced by weighting sums towards k-space. In fact, one finds

$$T_{opt} = \frac{4\pi}{3} \left(\frac{p}{\pi} \right)^{3/2} t_{pair} N^{3/2} \tag{38}$$

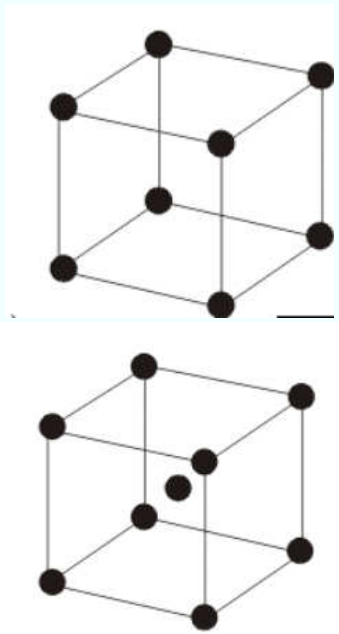
where p is desired precision (error $\propto \exp(-p)$) and t_{pair} computation time for single interaction pair.
For $p = \pi^2$, get

$$\begin{aligned} R_{opt} &= \pi^{1/2} L N^{-1/6} \\ \alpha L &= \pi^{1/2} N^{1/6} \end{aligned} \tag{39}$$

See article by Fincham (1994) for details.

8 – Assignment A3: Ewald summation

- For this assignment you can either complete the Ewald summation programm you started in last week's hands-on session or take the provided script ewald.py. Either way, the purpose is to compute the potential energy of a static, periodic, particle distribution.
- Set up a random particle distribution (of eg 100 particles) within a unit cube (include the z-component this time). Ensure that the seed used for the random number generator is fixed (and not derived, eg, from the clock). Check that your particle coordinates are reproduced in successive runs.
- Now examine the convergence behaviour of the real- and k-space parts of the summation respectively, for αL values (0.5,5,50), while varying n_{max} and H^2_{max}. Illustrate your results graphically.
- Madelung constant of ionic crystal. Now set up a crystal lattice with ions arranged in a simple-cubic (charges alternating) or body-centred cubic array (+ve charge surrounded by -ve ions) - see figure. Determine the 'sweet spot' (combination of αL and H^2_{max} for which just one real-space box ($n = 1$) is needed to compute the potential energy with a *relative* precision of 10^{-4} (see 16). The Madelung constant M of the lattice is defined by $U = -N_c q^2/aM$, where N_c is the number of ions per unit cell; a is the inter-ion spacing. (Generally accepted values are 1.748 for NaCl and 1.763 for CsCl.)



- M. J. L. Sangster & M. Dixon, *Interionic potentials in alkali halides and their use in simulations of the molten salts*, Advances in Physics **25**, 247–342 (1976)
- D. M. Heyes, *Electrostatic potentials and fields in infinite point charge lattices*, J. Chem. Phys. **74**, 1924 (1981)
- D. Fincham, *Optimisation of the Ewald sum for large systems*, Molecular Simulation **13**, 1–9 (1994)
- J. W. Perram, H. G. Petersen & S. W. De Leeuw, *An algorithm for the simulation of condensed matter which grows as the 3/2 power of the number of particles*, Mol. Phys. **65**, 875–893 (1988)

Part III

Particle-Mesh Methods

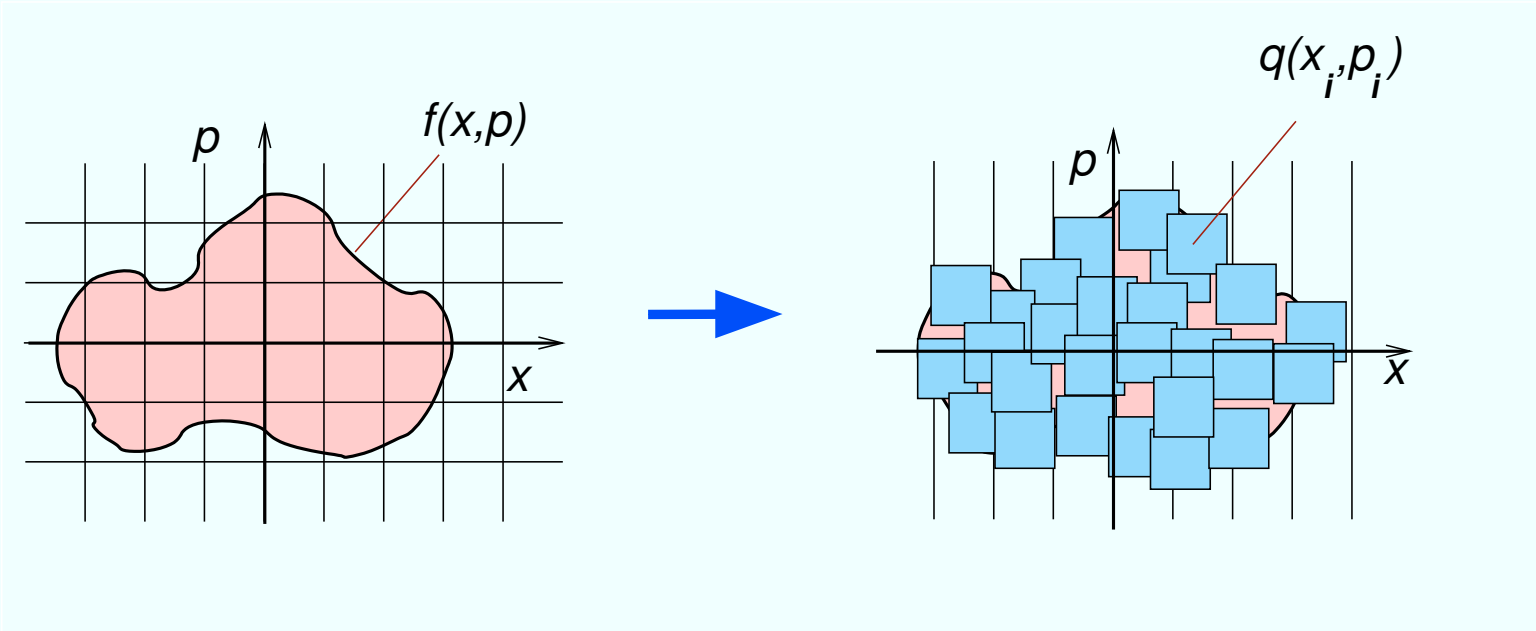
9 – Outline

Important method developed in the 1960s is the so-called Particle-Mesh (PM) technique. This was motivated by a need to handle kinetic effects in plasmas and astrophysical problems with good statistics, avoiding artificially high collisionality caused by graininess (low N). Basic idea: solve Vlasov-Boltzmann-like equation for single-particle velocity distribution function $f(\boldsymbol{x}, \boldsymbol{v})$ of discrete phase-space elements (charges):

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + q\boldsymbol{E} \cdot \frac{\partial f}{\partial \boldsymbol{v}} = \left.\frac{\partial f}{\partial t}\right|_c . \tag{40}$$

- Distribution function $f(\boldsymbol{x}, \boldsymbol{v})$ is 6-dimensional – direct solution of Eq. (40) possible but generally impractical.
- Collision term non-trivial: has to calculated from kinetic theory/particle dynamics!

Distribution function is represented instead by a large number of discrete *macro-particles*, each carrying a fixed charge q_i and mass m_i.



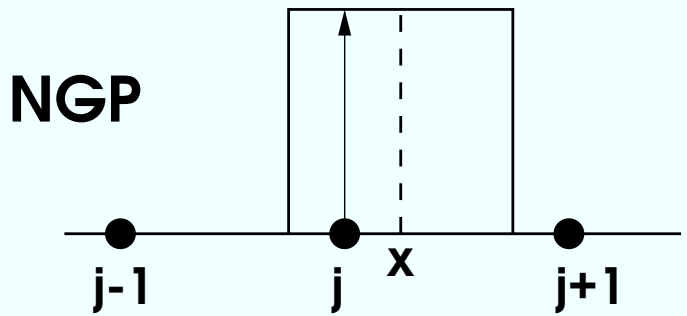
For macroscopic problems, ‘charges’ represent packets of large numbers of real particles, each carrying individual charge q_i such that the total charge matches that of the configuration in question, but with q_i/m_i chosen such that their motion obeys the physically consistent Lorentz/Newton equation:

$$\frac{d\boldsymbol{v}_i}{dt} = \frac{q_i}{m_i}\boldsymbol{E}_i(x,t), \quad i = 1...N \tag{41}$$

Instead of solving $\boldsymbol{E}_i$ via summation over the particles the charges are first mapped onto a grid to obtain a charge density:

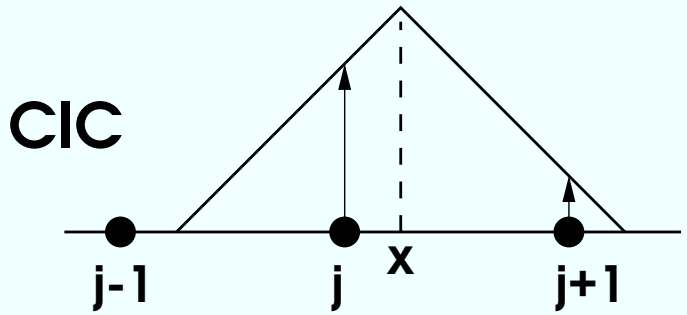
$$\rho(\boldsymbol{x}_j) \quad = \quad \sum_i q_i S(\boldsymbol{x_j} - \boldsymbol{x}_i), \qquad\qquad i = 1...N_{\text{cell}} \tag{42}$$

where $S(\boldsymbol{x_j} - \boldsymbol{x})$ is a function describing the effective shape of the particles. Usually it is sufficient to use a linear weighting for S — the 'Cloud-in-Cell' scheme — although other higher-order interpolations (eg: quadratic or cubic splines, Gaussian shape functions) are also possible.



Assuming uniform grid spacing $\Delta = x_j - x_{j-1}$, have weighting function:

$$S_1(x_j - x) = \begin{cases} 1, & | x_j - x | / \Delta < 1 \\ 0, & \text{otherwise} \end{cases}$$



Weighting function

$$S_2(x_j - x) = \begin{cases} 1 - \frac{|x_j - x|}{\Delta}, & | \, x_j - x \, | \, / \Delta < 1 \\ 0, & \text{otherwise} \end{cases}$$

Particle cloud has effective width 2Δ.

Density gather

```
weight = q/dx                  charge weighting factor

for i in range(npart):
      xa = x[i]/dx                 normalised coordinate
      j1 = int(xa)                 grid index
      j2 = j1 + 1
      f2 = xa - j1
      f1 = 1.0 - f2
      rho[j1] = rho[j1] + weight*f1     Accumulate charge
      rho[j2] = rho[j2] + weight*f2
```

In the electrostatic approximation, ($\boldsymbol{B} = 0; \nabla \times \boldsymbol{E} = 0$), the only Maxwell equation we need is Poisson's equation:

$$\nabla^2 \phi = \frac{\rho}{\varepsilon_0}. \tag{43}$$

Here, ρ is the net charge density summed over all particles. The electric field is just obtained from

$$\boldsymbol{E} = -\nabla \phi.$$

Depending on the problem geometry, Eq.43 can be solved either directly or by Fourier methods – see Hockney & Eastwood, Ch. 6.

- Low-dimensional problems: direct quadrature or 2nd order difference $\Rightarrow$ tridiagnonal matrix inversion:

$$\phi_{j+1} - 2\phi_j + \phi_{j-1} = -\Delta x^2 \frac{\rho_j}{\varepsilon_0}.$$

- Fourier space (periodic systems)

$$\begin{aligned} \tilde{\rho}(k) &= \mathcal{F}\{\rho(r)\} \\ k^2\tilde{\phi}(k) &= -\frac{\tilde{\rho}(k)}{\varepsilon_0} \\ \phi(r) &= \mathcal{F}^{-1}\{\tilde{\phi}(k)\} \end{aligned} \tag{44}$$

Exercise

Implement a field solver for the 1D Poisson equation using Fourier transforms:

$$\frac{d^2\phi(x)}{dx^2} = \rho(x)$$

Assume a density profile $\rho(x) = \rho_0 \sin 2\pi x/L$, discretised on a regular grid with periodic boundaries: $\rho(x \pm L) = \rho(x)$. Check your solution by computing E and ρ from $\phi(x)$.

To compute the gridded fields at the particle positions, we use the same weighting scheme as for the charge assignment (Eq. 42):

$$\boldsymbol{E}(\boldsymbol{x}_i) = \sum_j S(\boldsymbol{x}_i - \boldsymbol{x}_j)\boldsymbol{E}(\boldsymbol{x}_j). \quad (45)$$

Field interpolation grid to particle

```
for i in range(npart):
      xa = x[i]/dx
      j1 = int(xa)
      j2 = j1 + 1
      b2 = xa - j1
      b1 = 1.0 - b2
      exi = b1*Ex[j1] + b2*Ex[j2]
```

Advantages:

- Very fast field-solver (FFTs): main computational effort shifted to integration of particle motion (for dynamic problems).

Drawbacks:

- Spatial resolution limited to $l \geq \Delta$, mesh spacing: close range interactions such as binary encounters are inevitably smoothed out.
- For this reason, pure particle-mesh codes are normally used to treat for long-range, collective phenomena rather than molecular/atomistic systems.
- However, it is possible to correct short-range component with direct summation over neighbouring particles – basis of P3M method to be examined next.

10 – Outline

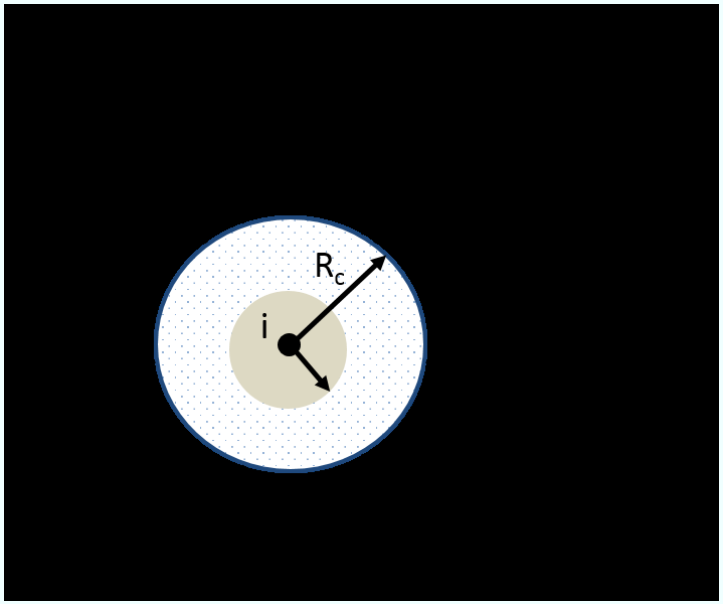
To improve the accuracy of fast PM method, one can consider compromise between brute-force direct sum and mesh-based field solver:

$$\boldsymbol{E}_{ij} = \boldsymbol{E}_{ij}^{PP} + \boldsymbol{E}_{ij}^{PM}, \tag{46}$$

where $\boldsymbol{E}_{ij}^{PP}$ is computed over several interparticle spacings up to a cutoff radius R_c (cf. 7), and $\boldsymbol{E}_{ij}^{PM}$ is computed on the mesh. Problems:

- Choice of R_c: too big $\rightarrow O(N^2)$ algorithm; too small $\rightarrow O(N)$, but low accuracy/resolution
- Need to match field and potential across cutoff sphere.

Consider NGP scheme – see Eq.2. This is equivalent to replacing point charges with spheres of radius $a/2$, volume $\pi a^3/6$.



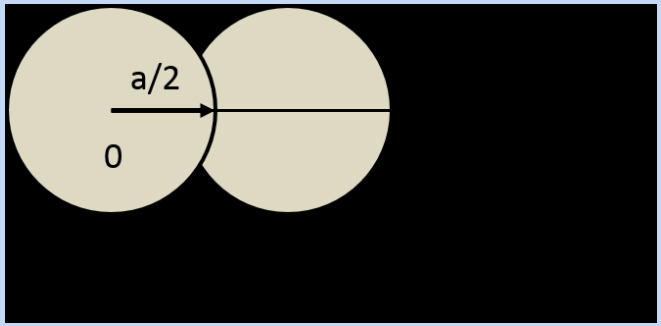
Density

$$\rho(r) = \begin{cases} \dfrac{6q}{\pi a^3}, & r < a/2 \\ 0, & r \geq a/2 \end{cases} \quad (47)$$

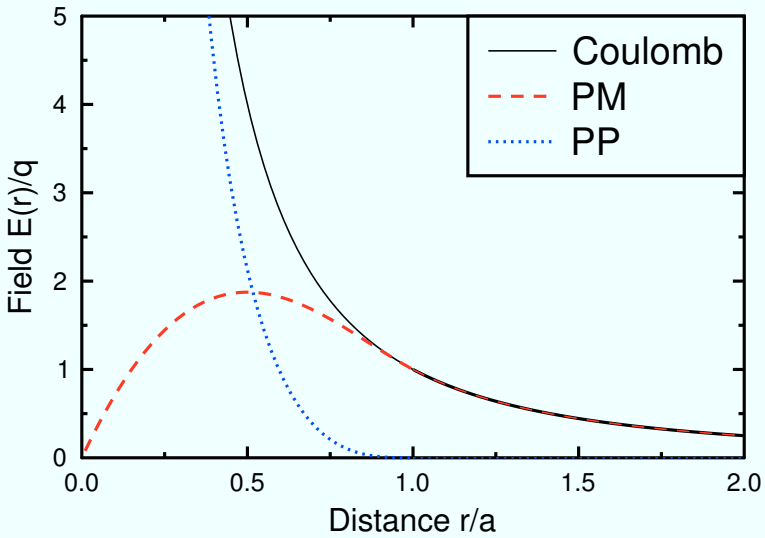
Exercise

Elementary electrostatics (Gauss' law) shows that the force between two such spheres along the axis joining their centres is given by:

$$E_s(r) = \begin{cases} \dfrac{q^2}{a^2}\left(\dfrac{8r}{a} - \dfrac{9r^2}{a^2} + \dfrac{2r^4}{a^4}\right), & r < a \\ \dfrac{q^2}{r^2}, & r \geq a \end{cases} \tag{48}$$



The natural splitting choice in this case is to take the short-range cutoff $R_c = a$, and to set the PP field inside this sphere equal to the *difference* between the Coulomb field and this effective field contribution arising from the mesh points:



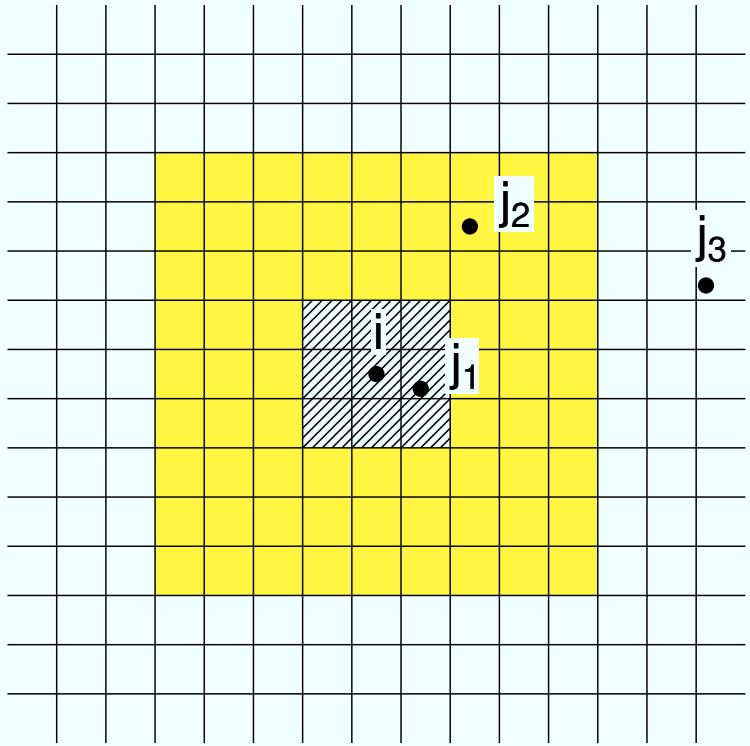
$$E^{PP}(r) = \begin{cases} E_c(r) - E_s(r), & r < a \\ 0, & r \geq a \end{cases}$$

The exact expression for the PM field at every particle i is:

$$E_i^{PM} = \sum_j E_s(r_i - r_j), \tag{49}$$

which would of course take an effort $O(N^2)$ to evaluate – back to square one! This is the algorithmic challenge of P3M: to design a fast, approximate PM calculation which minimises the difference between fields evaluated on the mesh and the exact reference field of Eq.49.
In fact the PM scheme offers presents opportunities to introduce corrections at each step in the cycle – see Hockney & Eastwood, Chap. 8. NGP is usually replaced by other, better performing shape functions – Arnold, FZJ Lecture Notes, pp.46–51.

Cubic cutoff – (Nishihara, 1991): For each cell do direct sum for near neighbours, then PM calculation for remainder excluding PP charge inside central region.



11 – Outline

Developed by Darden, 1993 combining force-splitting idea of P3M, but retaining physical resemblance to original Ewald sum through Gaussian spread function. Main motivation is to shift burden of computation towards k-space, then use FFTs to evaluate this part.

Taking a large number of k-vectors per particle implies:

$$N_k \sim \frac{N}{nR_c^3},$$

where n is the number density. Instead of summing these directly, PME maps charges onto a grid (like in the PM method) and uses FFTs to speed up the k-space computation. With a fixed number of particles per cell, the scheme is then $O(N \log N)$, compared to the best possible direct Ewald of $O(N^{3/2})$.

Recall k-space sum from Eq.21:

$$V_E^k(r_i) = \frac{4\pi}{L^3} \sum_{\boldsymbol{k}\neq 0} \sum_j q_j \frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{|\ \boldsymbol{k}\ |^2} \exp\left\{i\boldsymbol{k}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)\right\}$$

for corresponding smeared charge densities:

$$\rho'(r) = \sum_j q_j \sigma(r - r_j)$$

By inspection of (29), we can therefore write down the Fourier transformed density on the grid:

$$\tilde{\rho}_{jkl}(k) = \frac{1}{L^3} \exp\left(-\frac{k^2}{4\alpha^2}\right) \sum_i q_i \exp(i\boldsymbol{k}\cdot\boldsymbol{r}_i) \tag{50}$$

The potential in k-space is then just the density multiplied by an 'influence' function $\tilde{G}(k)$:

$$\tilde{\phi}_{jkl}(k) = \tilde{G}(k)\tilde{\rho}(k), \tag{51}$$

where

$$\tilde{G}(k,\alpha) = \frac{\exp(-k^2/4\alpha^2)}{k^2}, \tag{52}$$

and the fields are given simply by:

$$\begin{aligned} \tilde{E}_{jkl}(\boldsymbol{k}) &= -i\boldsymbol{k}\tilde{\phi}_{jkl}(k) \\ &= -i\boldsymbol{k}\tilde{G}(k,\alpha)\tilde{\rho}(k). \end{aligned} \tag{53}$$

The inverse FT then yields the fields and potential at the mesh points, which in turn can be interpolated back to the particle positions.

- Disadvantage of PME over standard P3M: due to choice of Gaussian smearing function, large number of grid points needed per particle (typically $8^3 \sim 500$) $\rightarrow$ bottleneck in mapping to/from grid. Cf: 8 for CIC; 27 for TSC shape functions in P3M.
- Apart from choice of smearing function, PME is effectively a variation of P3M, but was developed independently.
- Modern P3M generally faster than original PME.

11 – Hands-on session HO2: practice with P3M techniques

- Download the script `fft-poisson-solver.py` from the script folder on Toledo. This provides a solution to the exercise on 79. First verify that the computed $\phi(x)$ really satisfies $\nabla^2\phi = \rho$. If you are not familiar with the numerical implementation of DFTs, it is worth spending some time inspecting the form and numerical representation of $\rho(k), \phi(k)$ etc. (Note that here, a complex FT is computed from a real input: care needs to be taken before performing the IFTs). This procedure is readily generalised to 2- and 3D.
- Modify this script to compute a piecewise FFT from two or more segments of the density. eg: let $\rho(x) = \rho_1(x) + \rho_2(x)$ and compute the corresponding potentials ϕ_1, ϕ_2 from the partial densities. Check that $\phi_1 + \phi_2 = \phi$, the potential from part 1.
- Now download the particle-mesh code pm-1d.py provided. This is a fully functional 1D electrostatic particle-mesh code, solving Poisson's equation together with the particle equations of motion in one spatial and one velocity dimension. Currently it is set up to follow the evolution of a plasma density perturbation (Langmuir wave) over a few periods. Experiment with the setup (number of particles, wave amplitude, wavelength) to get a feel for the diagnostic output and plasma wave behaviour.
- The field solver used is a simple Trapezium integration (possible only in 1D). Replace this with the FFT algorithm you developed above - ie: compute ϕ first and derive E_x from $d\phi/dx$. This will serve as the long-range mesh part in the final P3M code.

11 – Hands-on session HO2: practice with P3M techniques contd

- As explained in the notes, the pure PM algorithm smooths over fluctuations with wavelengths below the grid spacing. This can be corrected by evaluating the short-range part directly, but to do this we need a way of determining near neighbours. This can be done with the following linked list algorithm (head≥ 0 points to the first particle in the cell; link(i) points to the next in the list, until -1 is reached)

```
for j in range(ngrid)
    head(j) = -1
for i in range(N)
    j=int(x(i)/dx)
    list(i)=head(j)
    head(j)=i
```

- Implement the above algorithm in the PM code and verify that it picks out sets of particles belonging to each grid cell.
- Now combine the piecewise FFT technique with the linked list algorithm to construct a P3M algorithm analogous to the method used by Nishihara.
  - For each set of 3 grid points, use FFTs to compute the far-field E_{PM} of the charge distribution excluding these cells.
  - Use the linked lists to compute short-range fields E_{PP} of the particles in the 3 neighbour cells.
  - Combine the results $E_x = E_{PM} + E_{PP}$ and compare with the original pure mesh-based field solution. Comment on the differences and test the energy/momentum conservation of the P3M system.

- R. W. Hockney & J. W. Eastwood, *Computer Simulation using Particles*, Taylor & Francis, New York (1988).
- T. Darden, D. York, and L. Pedersen, Particle Mesh Ewald: An $Nlog(N)$ method for Ewald sums in large systems, J. Chem. Phys., 98, 10089–10092, 1993.
- U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pedersen, A smooth Particle Mesh Ewald method, J. Chem. Phys., 103, 8577, 1995.
- M. Deserno and C. Holm, How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines, J. Chem. Phys., 109, 7678, 1998.

9 Particle-Mesh methods

10 P3M: Particle-Particle, Particle-Mesh

11 Particle-Mesh-Ewald

Part IV

Multipole methods 1: Tree codes

12 Tree codes

13 Basic tree algorithm

14 Multipole expansion

15 Shifting properties

16 Tree walk

17 Error control

18 Algorithm cycle

12 – Outline

Discretize charge density with point particles:

$$\rho(\boldsymbol{r}, t) \quad = \quad \sum_j q_j \delta(\boldsymbol{r}_j - \boldsymbol{r}). \tag{54}$$

Substituting (54) into (9) gives pairwise potential:

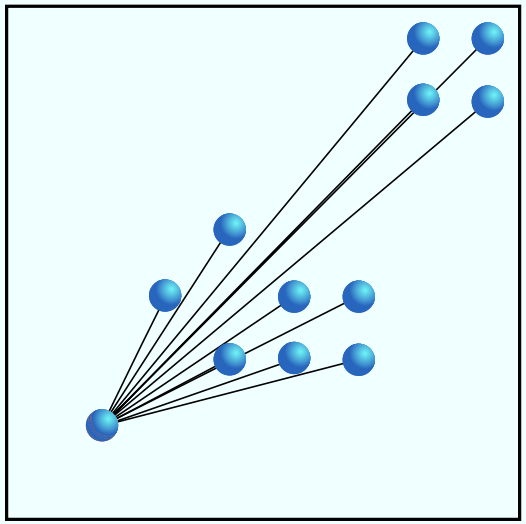
$$\phi_i(\boldsymbol{r}_i, t) \quad = \quad \sum_{j \neq i} \frac{q_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} \tag{55}$$

Dynamics governed by Newton's law:

$$\frac{d\boldsymbol{v}_i}{dt} = -\frac{q_i}{m_i} \nabla \phi(\boldsymbol{r}_i); \qquad \frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i. \tag{56}$$

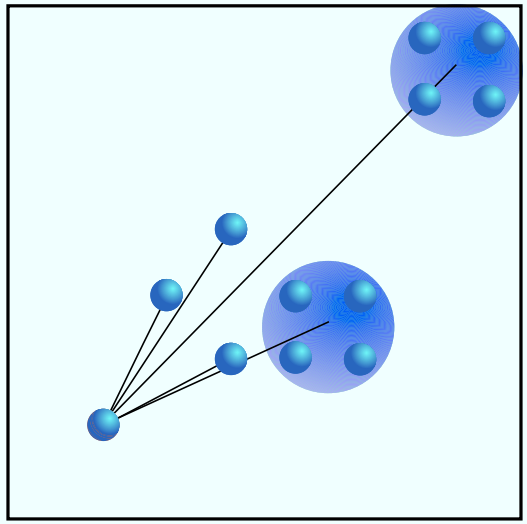
$\Rightarrow$ Basis for MD with charged particles, gravitational dynamics

Direct Summation



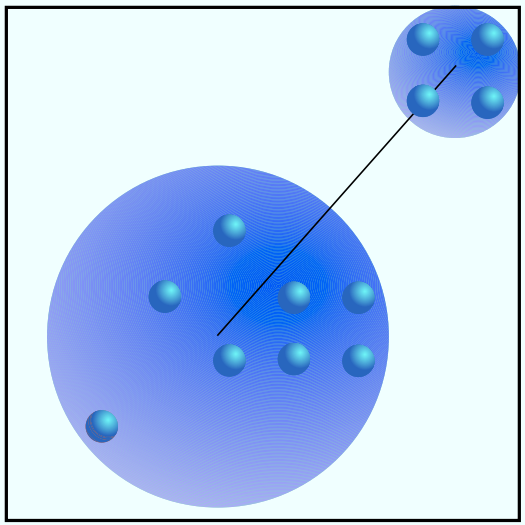
interactions: N^2

Treecode



$N \log N$

Fast Multipole Method

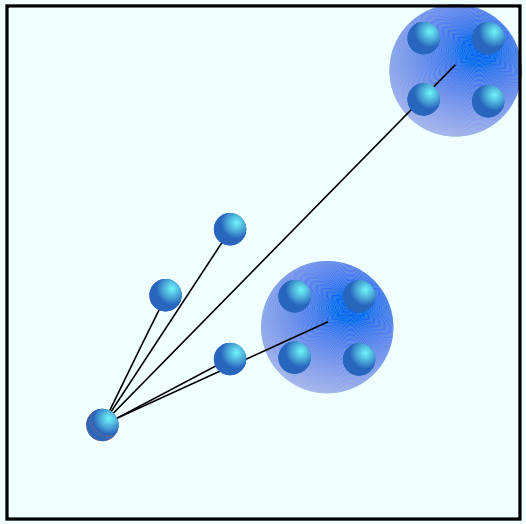


N

Barnes-Hut Tree Code
particle-cluster interactions $\rightarrow O(N \log N)$
J. Barnes & P. Hut, Nature (1986)

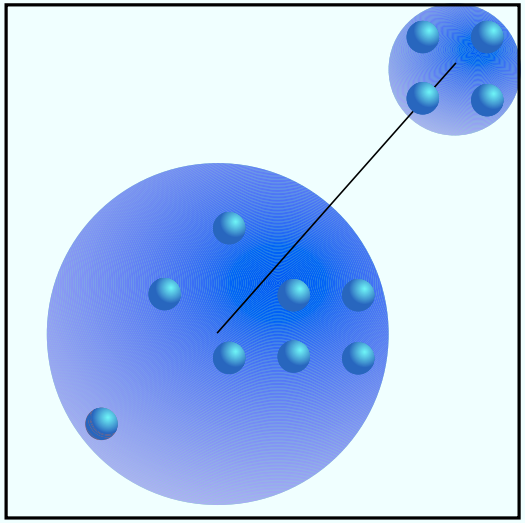
Fast-Multipole Method
cluster-cluster interactions $\rightarrow O(N)$
L. Greengard & V. Rohklin, J. Comp. Phys. (1987)

Treecode



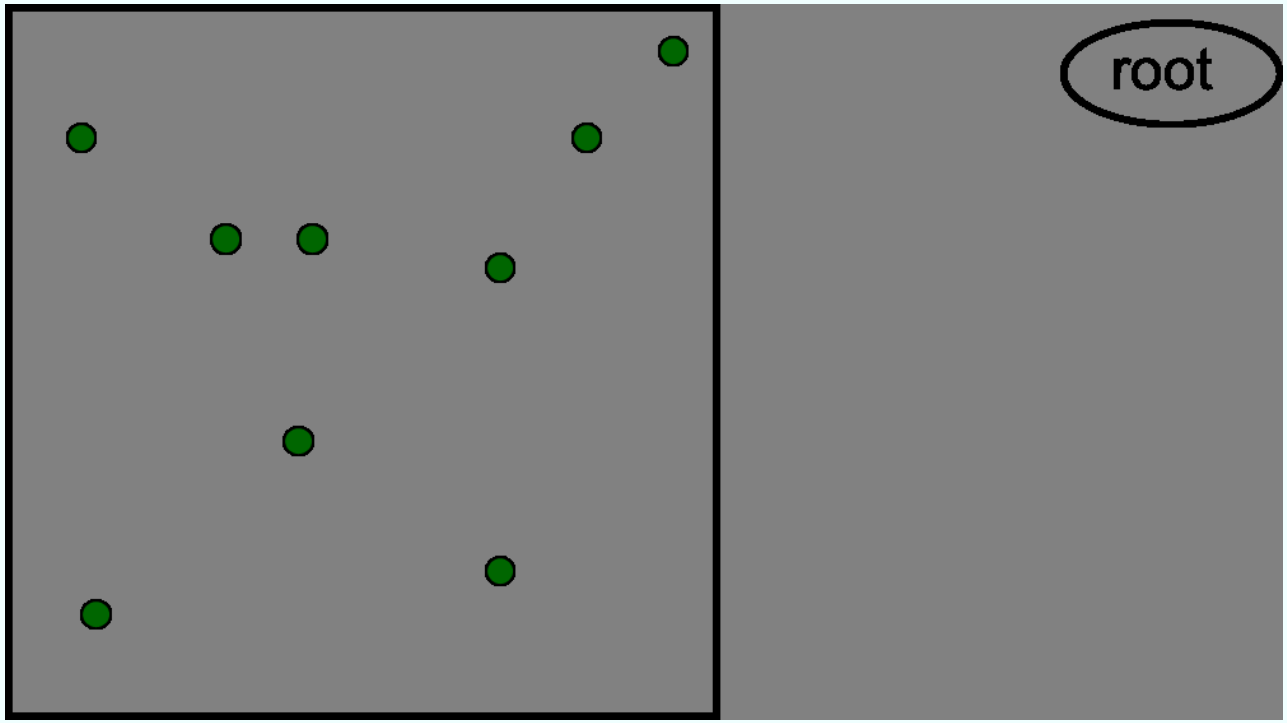
- $O(N \log N)$
- dynamic load-balancing
- versatility + adaptability
- easy exchange of interaction kernel

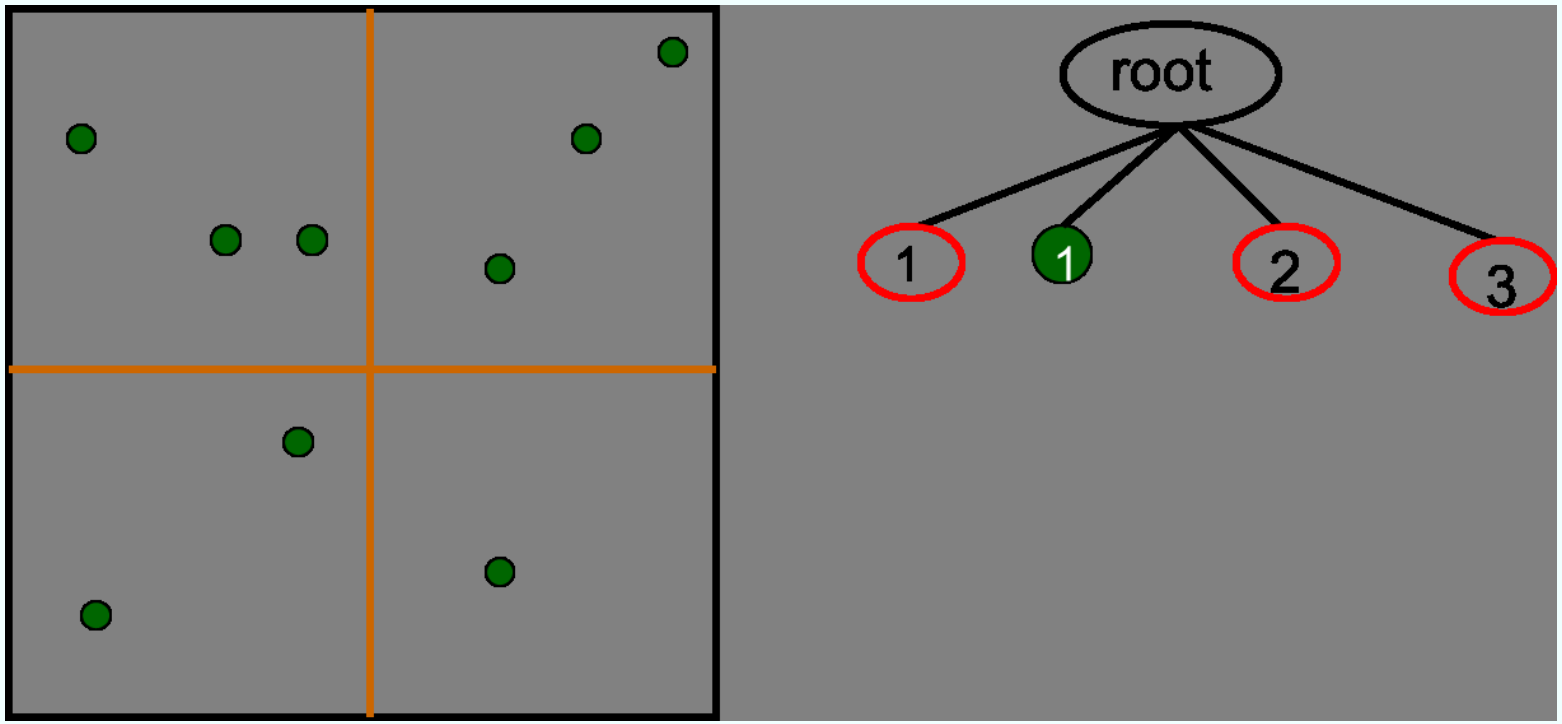
FMM

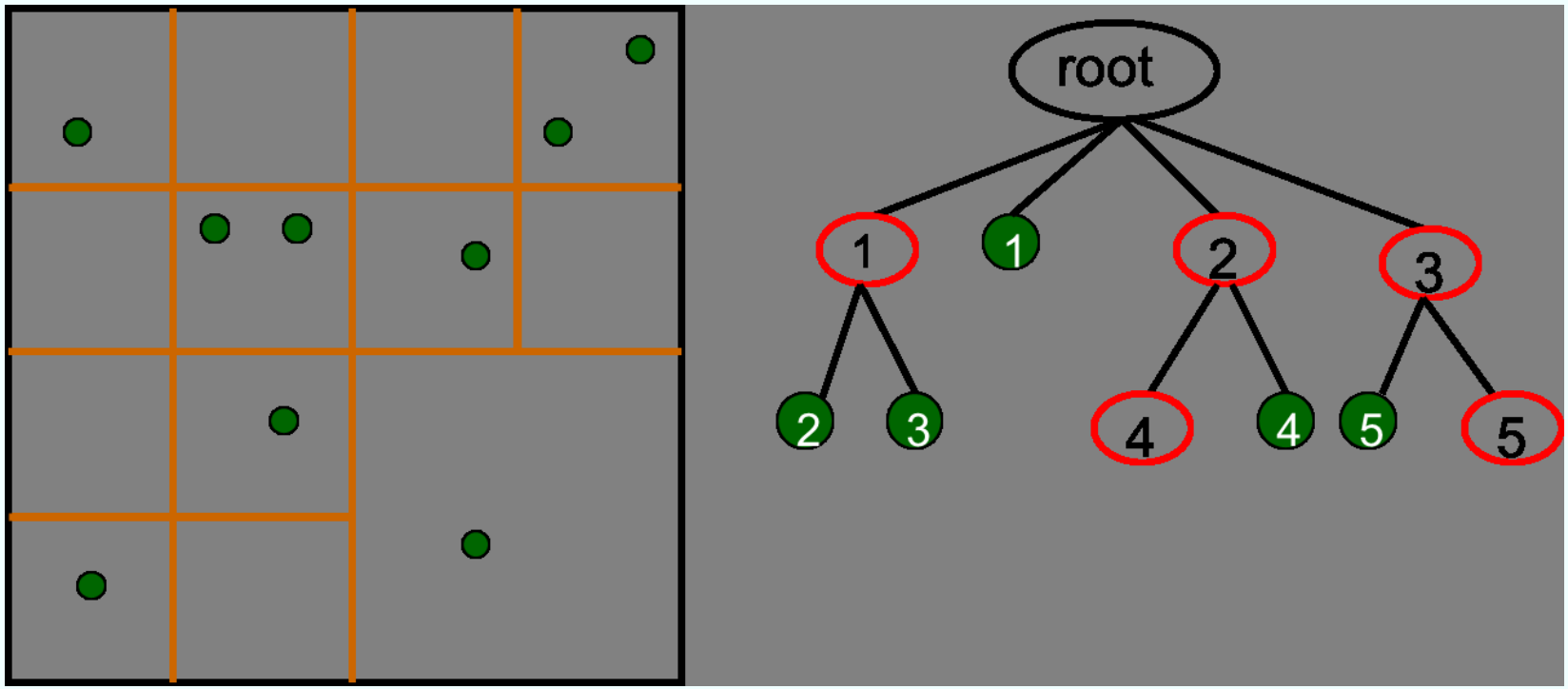


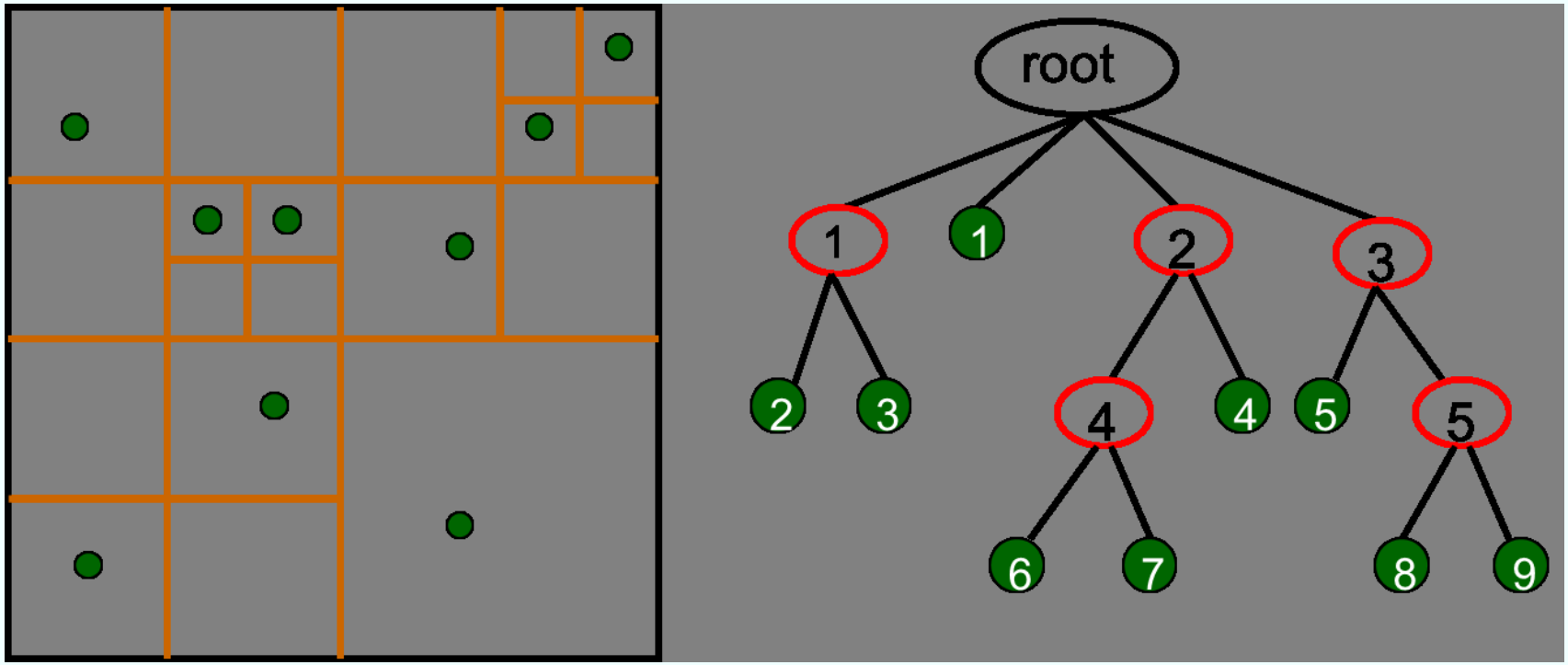
- $O(N)$
- rigorous error control
- high (machine) accuracy
- fast!

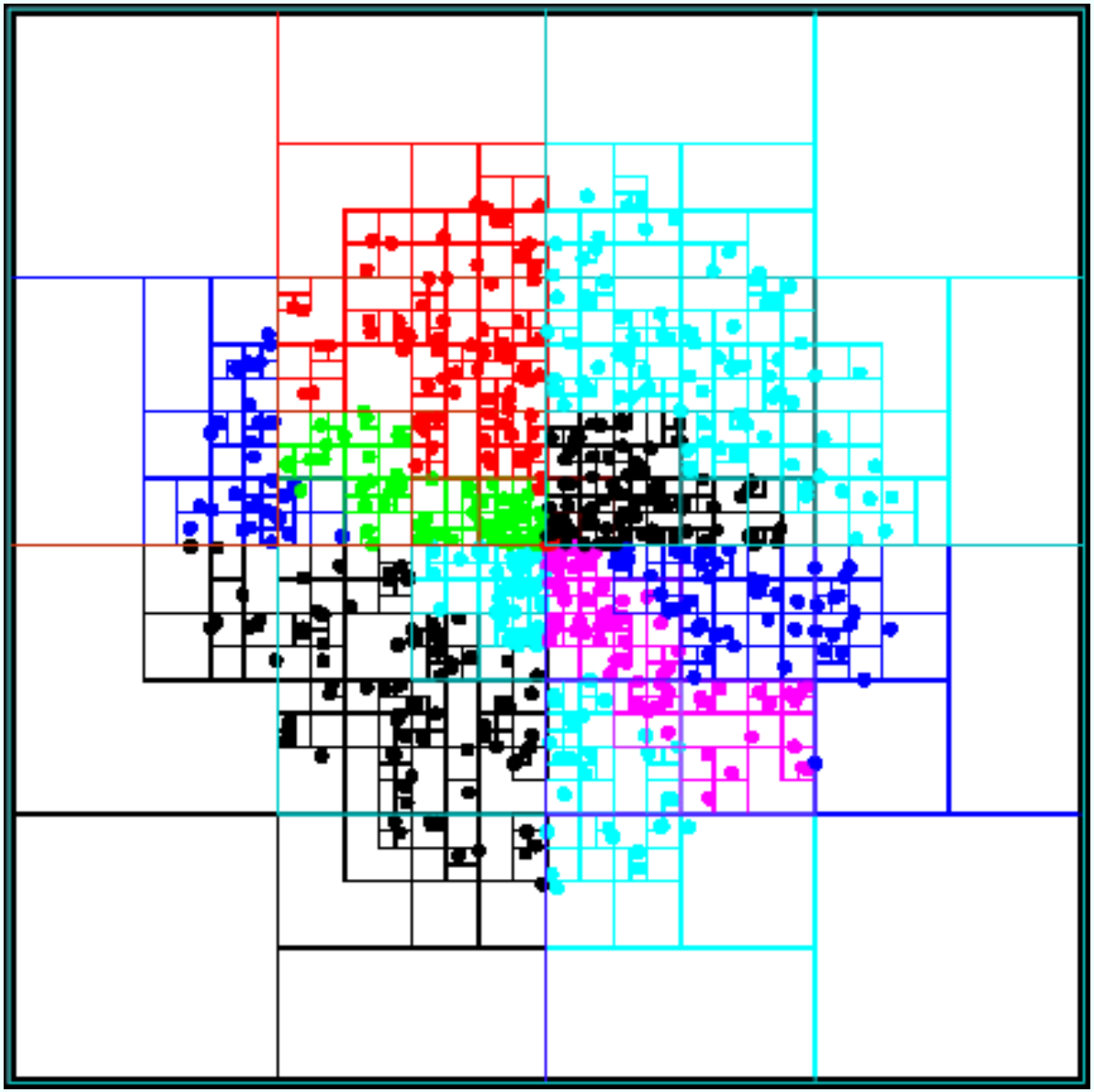
13 – Outline


root

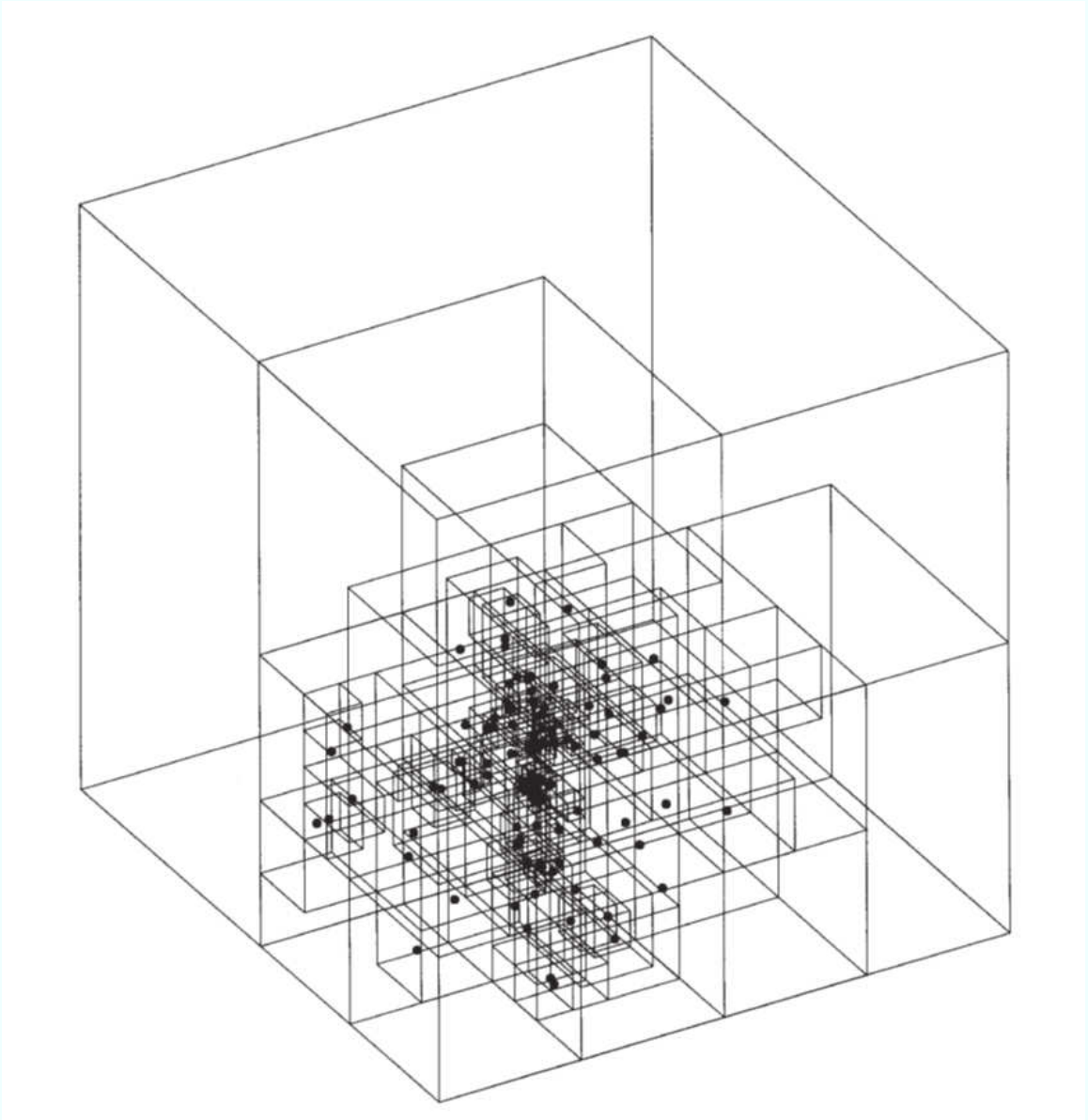

root
1
1
2
3

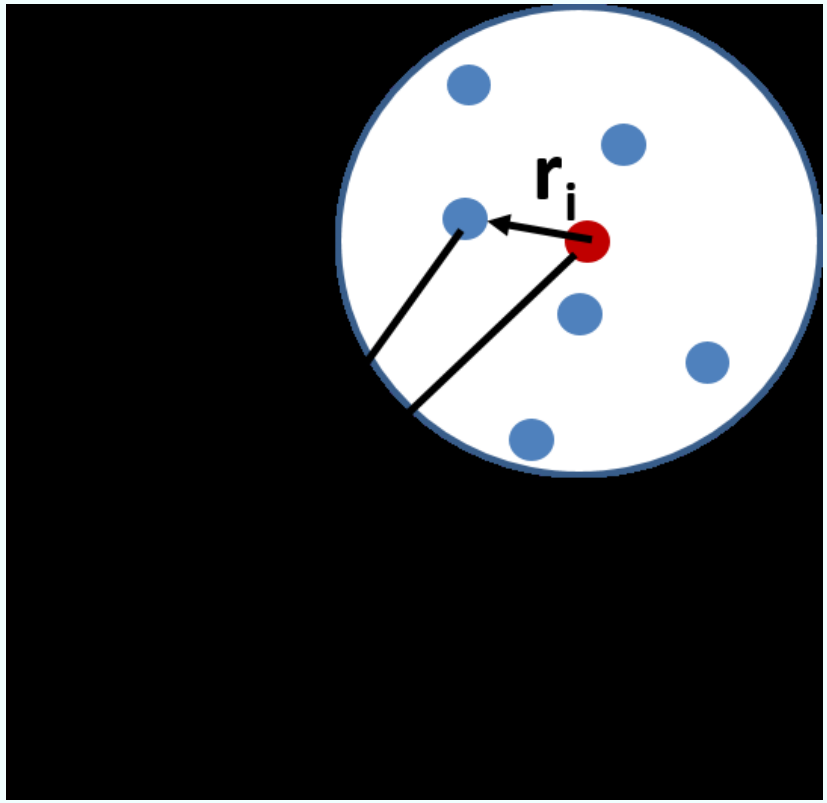

root
1
1
2
3
2
3
4
4
5
5


root
1
1
2
3
2
3
4
4
5
5
6
7
8
9



13 – 2) 3D tree structure for clustered mass distribution





$$\begin{aligned}\Phi_i(\mathbf{r}-\mathbf{r}_i) &= \frac{q_i}{|\mathbf{r}-\mathbf{r}_i|} \\ &= \frac{q_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}.\end{aligned}$$

Expanding this potential about location $\boldsymbol{r}$ (eg: centre of charge) gives:

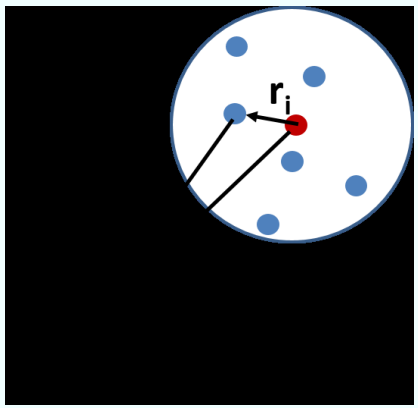
$$
\begin{aligned}
\Phi(\mathbf{r}) &= \sum_i q_i \left[1 - \mathbf{r}_i \frac{\partial}{\partial \mathbf{r}} + \frac{1}{2}\mathbf{r}\,\mathbf{r}\frac{\partial}{\partial \mathbf{r}}\frac{\partial}{\partial \mathbf{r}} + \ldots\right]\frac{1}{r} \\
&= \sum_i q_i \left[1 - x_i\frac{\partial}{\partial x} - y_i\frac{\partial}{\partial y} - z_i\frac{\partial}{\partial z}\right. \\
&\quad + \frac{1}{2}x_i^2\frac{\partial}{\partial x}\frac{\partial}{\partial x} + \frac{1}{2}y_i^2\frac{\partial}{\partial y}\frac{\partial}{\partial y} + \frac{1}{2}z_i^2\frac{\partial}{\partial z}\frac{\partial}{\partial z} \\
&\quad + \frac{1}{2}x_i y_i\left(\frac{\partial}{\partial x}\frac{\partial}{\partial y} + \frac{\partial}{\partial y}\frac{\partial}{\partial x}\right) \\
&\quad + \frac{1}{2}y_i z_i\left(\frac{\partial}{\partial y}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}\frac{\partial}{\partial y}\right) \\
&\quad \left. + \frac{1}{2}x_i z_i\left(\frac{\partial}{\partial x}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}\frac{\partial}{\partial x}\right)\right]\frac{1}{r} + O(r_i^3/r^4)
\end{aligned}
\tag{57}
$$

$$
\begin{aligned}
\Phi(\mathbf{r}) \quad = \quad & \sum_i q_i \left[\frac{1}{r} + x_i \frac{x}{r^3} + y_i \frac{y}{r^3} + z_i \frac{z}{r^3} \right. \\
& + \frac{1}{2} x_i^2 \left(-\frac{1}{r^3} + \frac{3x^2}{r^5} \right) + \frac{1}{2} y_i^2 \left(-\frac{1}{r^3} + \frac{3y^2}{r^5} \right) \\
& + \frac{1}{2} z_i^2 \left(-\frac{1}{r^3} + \frac{3z^2}{r^5} \right) \\
& \left. + x_i y_i \left(\frac{3xy}{r^5} \right) + y_i z_i \left(\frac{3yz}{r^5} \right) + x_i z_i \left(\frac{3xz}{r^5} \right) \right] \\
& + O(r_i^3 / r^4)
\end{aligned}
\tag{58}
$$

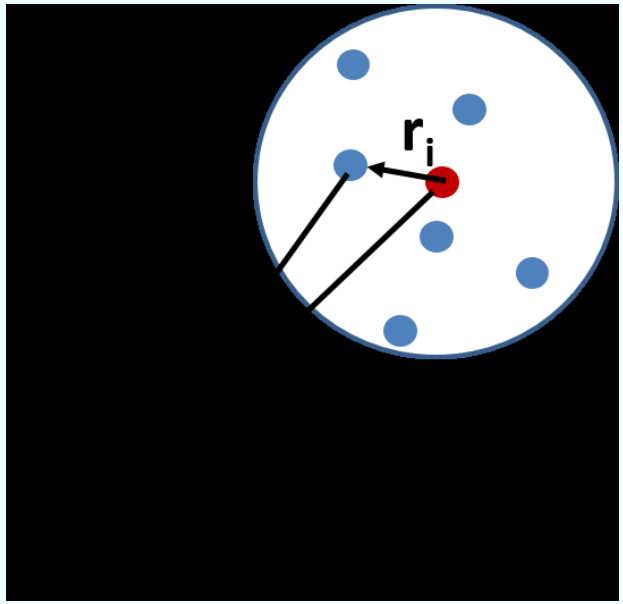
This (Eq.58) can be rearranged to give:

$$\Phi(\boldsymbol{r}) \quad = \quad \frac{M}{r} + \frac{r_\alpha D^\alpha}{r^3} + \frac{1}{2}\frac{r_\alpha r_\beta Q^{\alpha\beta}}{r^5}, \tag{59}$$

where M, D^α and $Q^{\alpha\beta}$ are the monopole, dipole and quadrupole moments of the particle clusters. Summation convention is assumed over indices α and β for coordinate components x, y, z. Also $\boldsymbol{r}_{i\alpha} = (x_i, y_i, z_i)$ refer to particle coordinates relative to cluster centre:



$$\boldsymbol{r}_c = \frac{\sum_i \mid q_i \mid \mathbf{r}_i}{\sum_i \mid q_i \mid} \tag{60}$$

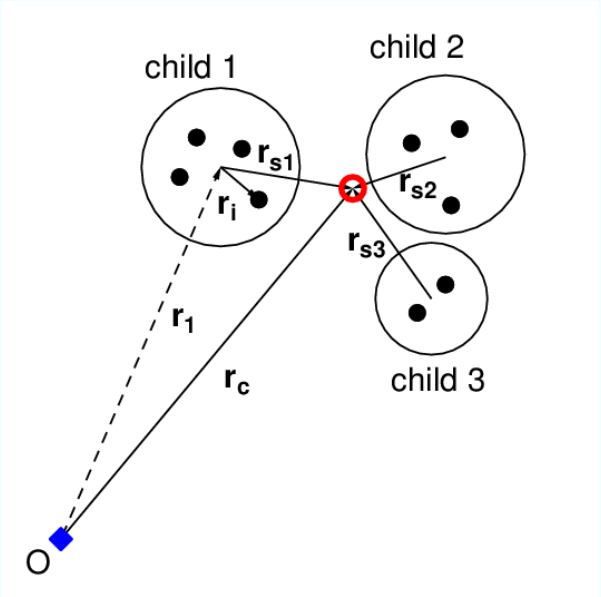


$$\Phi(\boldsymbol{r}) = \frac{M}{r} + \frac{r_\alpha D^\alpha}{r^3} + \frac{1}{2}\frac{r_\alpha r_\beta Q^{\alpha\beta}}{r^5} + ...; (\alpha, \beta = 1, 2, 3)$$

$$M = \sum_i q_i$$

$$D^\alpha = \sum_i q_i r_i^\alpha$$

$$Q^{\alpha\beta} = \sum_i q_i (3 r_i^\alpha r_i^\beta - r_i^2 \delta_{\alpha\beta}) \quad (61)$$



Shifting vector:

$$\boldsymbol{r}_c^{\text{parent}} = \frac{\sum_{\text{child-nodes}} \boldsymbol{r}_c^{\text{child}} \mid q_{\text{child}} \mid}{\sum_{\text{child-nodes}} \mid q_{\text{child}} \mid}$$

$$\boldsymbol{r}_s = \boldsymbol{r}_c(\text{child}) - \boldsymbol{r}_c(\text{parent}) \qquad (62)$$

Consider shifted dipole moment of single child node:

$$\begin{aligned} D^s_\alpha &= \sum_i q_i (r^\alpha_i - r^\alpha_s) \\ &= \sum_i q_i r^\alpha_i - r^\alpha_s \sum_i q_i \\ &= D_\alpha - r^\alpha_s M \end{aligned} \tag{63}$$

Similarly (for example),

$$\begin{aligned} Q^s_{xx} &= \sum_i q_i (x_i - x_s)^2 \\ &= \sum_i q_i x_i^2 - 2x_{sd} \sum_i q_i x_i + x^2_{sd} \sum_i q_i \\ &= Q_{xx} - 2x_s D_x + x_s^2 M \end{aligned}$$

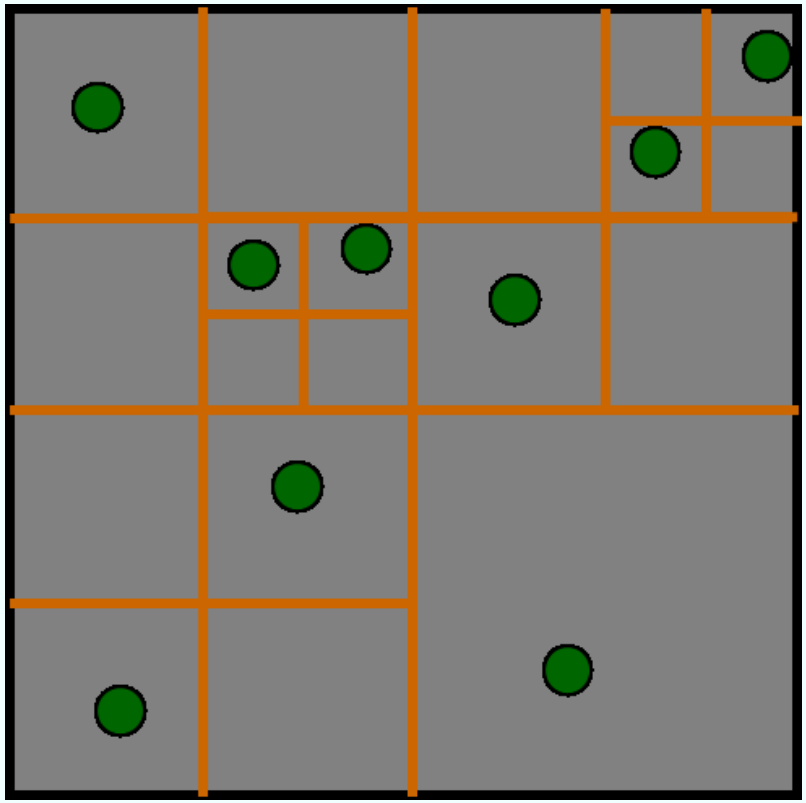
$$\text{and} \qquad Q^s_{xy} = Q_{xy} - x_s D_y - y_x D_x + x_s y_s M \tag{64}$$

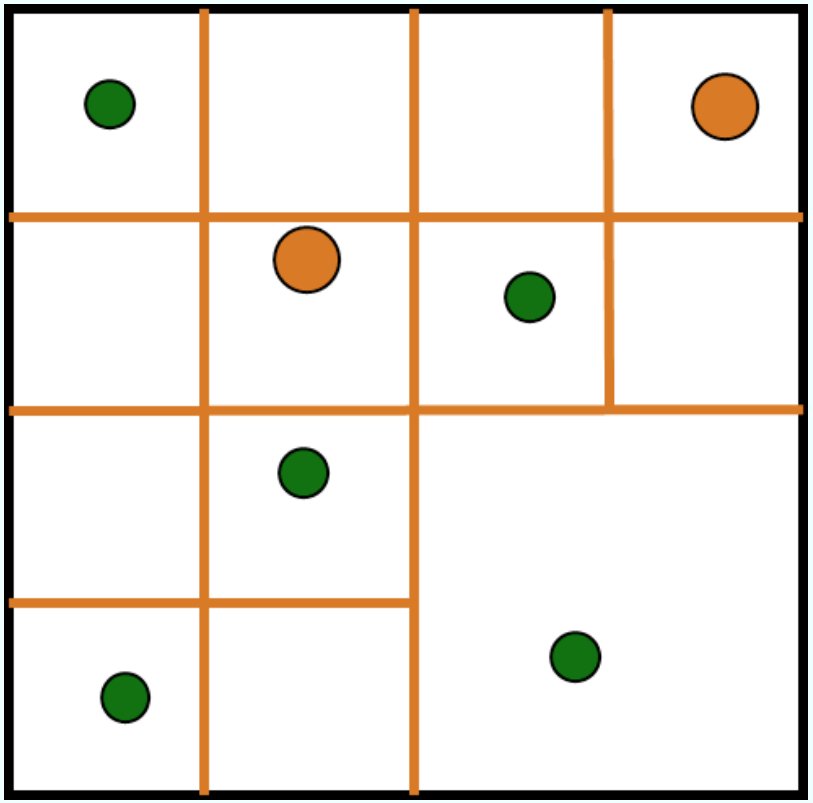
Important property of multipole algorithms:
All moments of the parent node can be obtained from knowledge of the child moments and the shifting vectors $\boldsymbol{r}_s$.

$$
\begin{aligned}
M^{\text{parent}} &= \sum_{\text{child-nodes}} M \\
D_\alpha^{\text{parent}} &= \sum_{\text{child-nodes}} D^s \\
&= \sum_{\text{child-nodes}} D_\alpha - r_s^\alpha M \\
Q_{\alpha\beta}^{\text{parent}} &= \sum_{\text{child-nodes}} Q_{\alpha\beta}^s = \ldots
\end{aligned}
\tag{65}
$$

Exercise

Complete the above shifting rules by deriving general expression for $Q_{\alpha\beta}^s$ starting from $Q_{\alpha\beta}$ in Eq.(61)

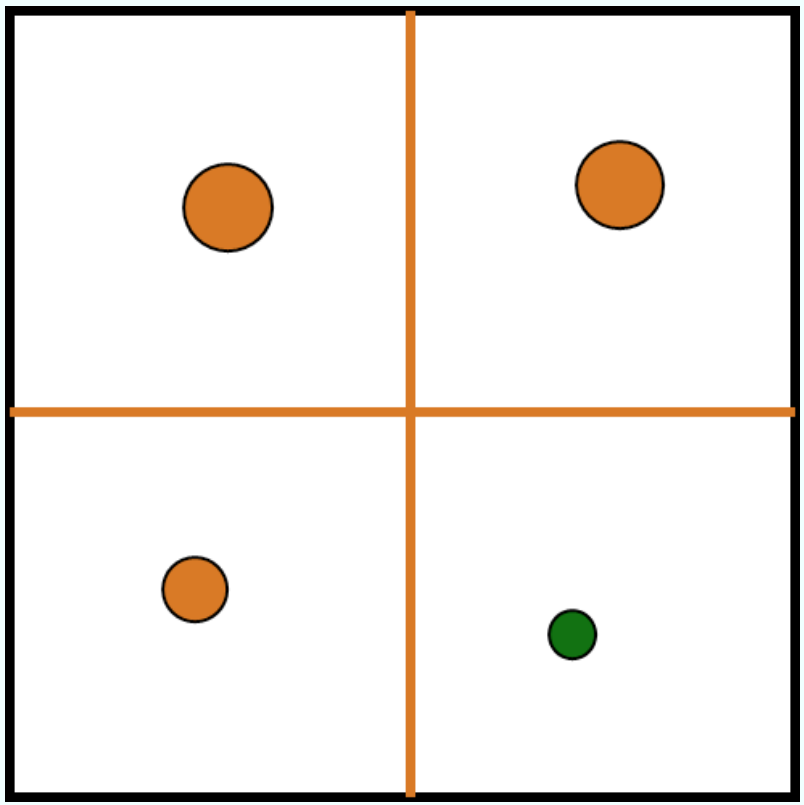


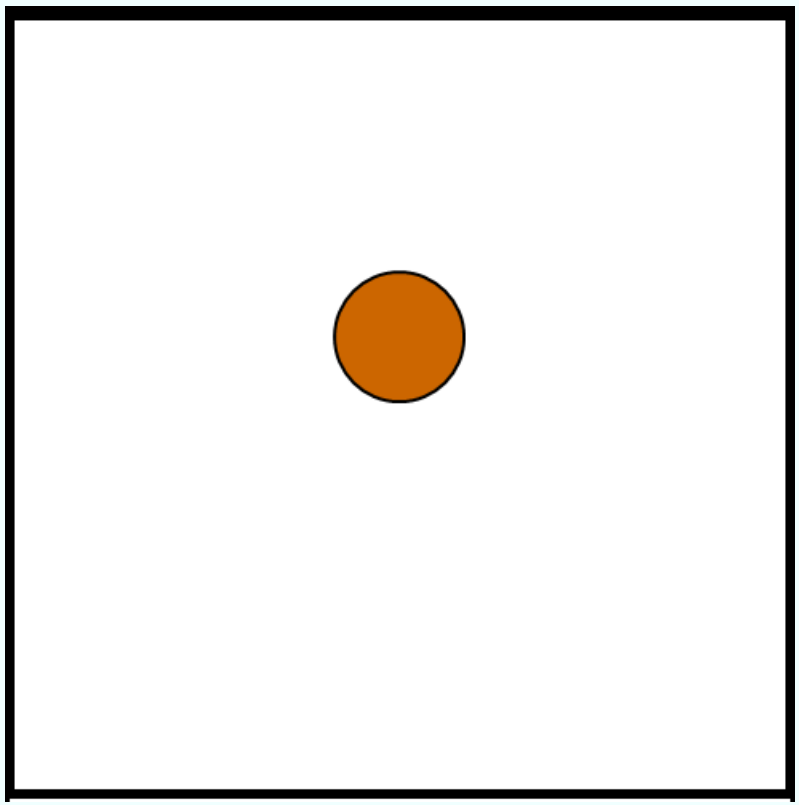


Multipole shifting properties:

$$D_\alpha^{\text{parent}} = \sum_{\text{child}} (D_\alpha - r_s^\alpha M)$$

etc.





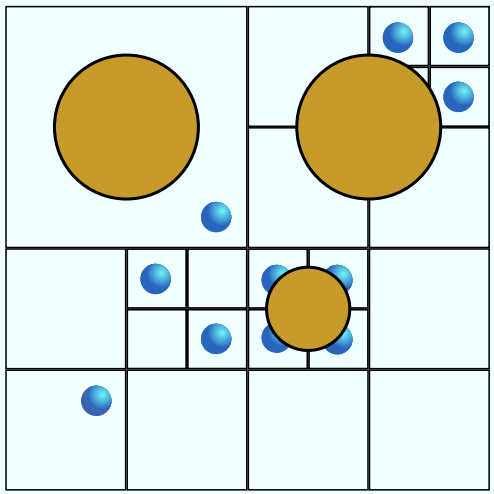
How many divisions are required to reach a typical tree node/cell? Average volume of cell containing one or more particles is the volume of the root cell V divided by the number of simulation particles N. Moreover, the average length of a cell is a power of 2 smaller than the root box:

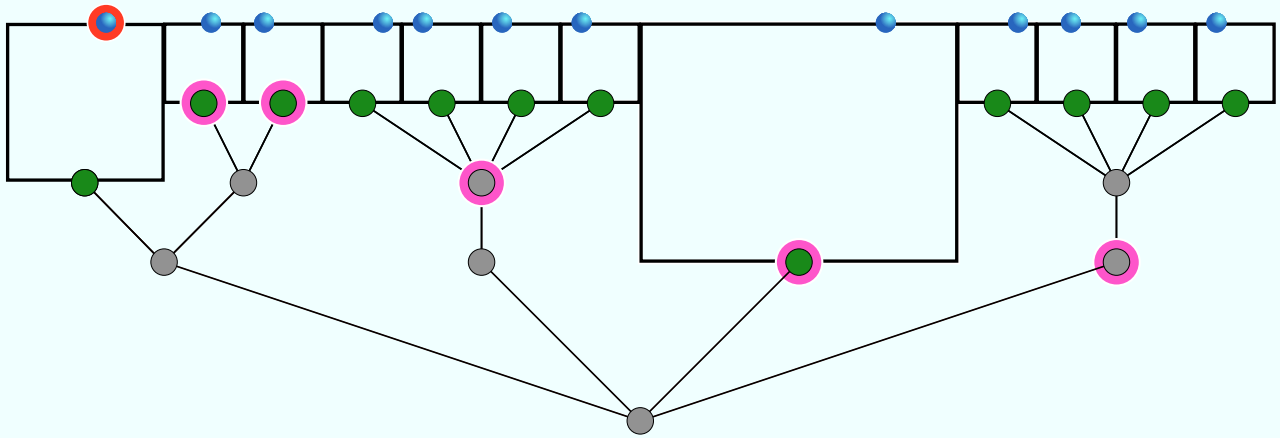
$$\left(\frac{V}{N}\right)^{1/3} = \frac{V^{1/3}}{2^h},$$

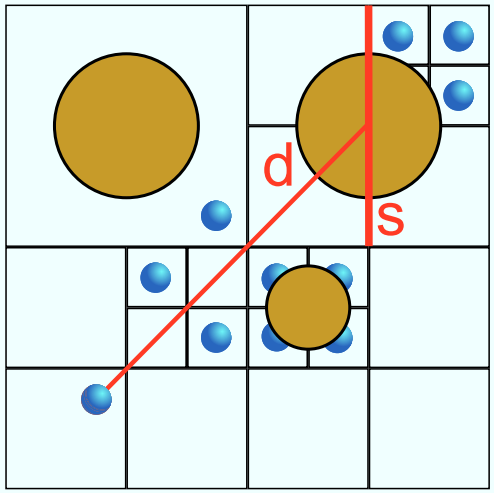
Therefore, height h of the tree is of the order

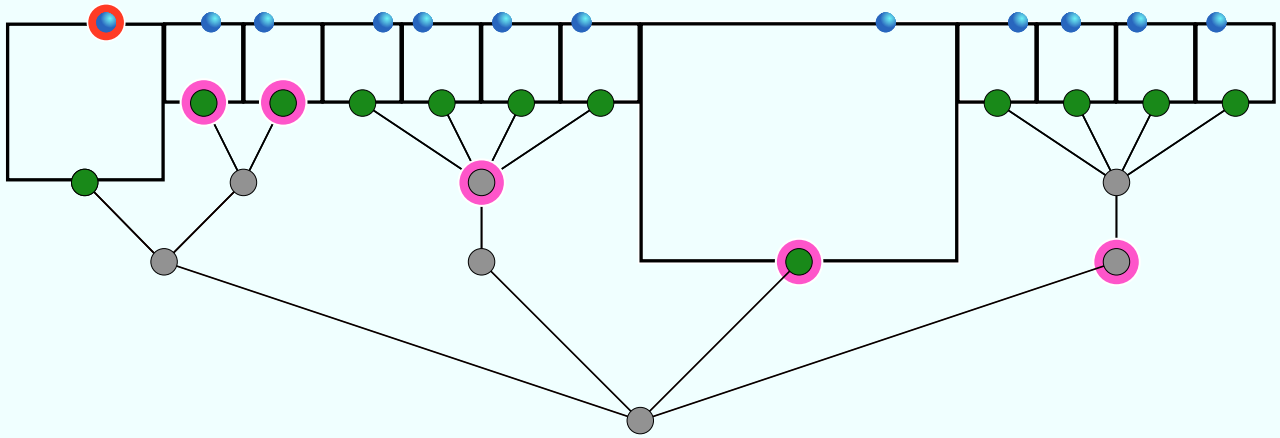
$$\log_2 N^{1/3} = \frac{1}{3\log 2}\log N \simeq \log N. \tag{66}$$

Starting from the root, an average of $\log N$ divisions are necessary to reach a given leaf. The tree contains N leaves, therefore the time required to construct the tree and its moments is $O(N \log N)$.





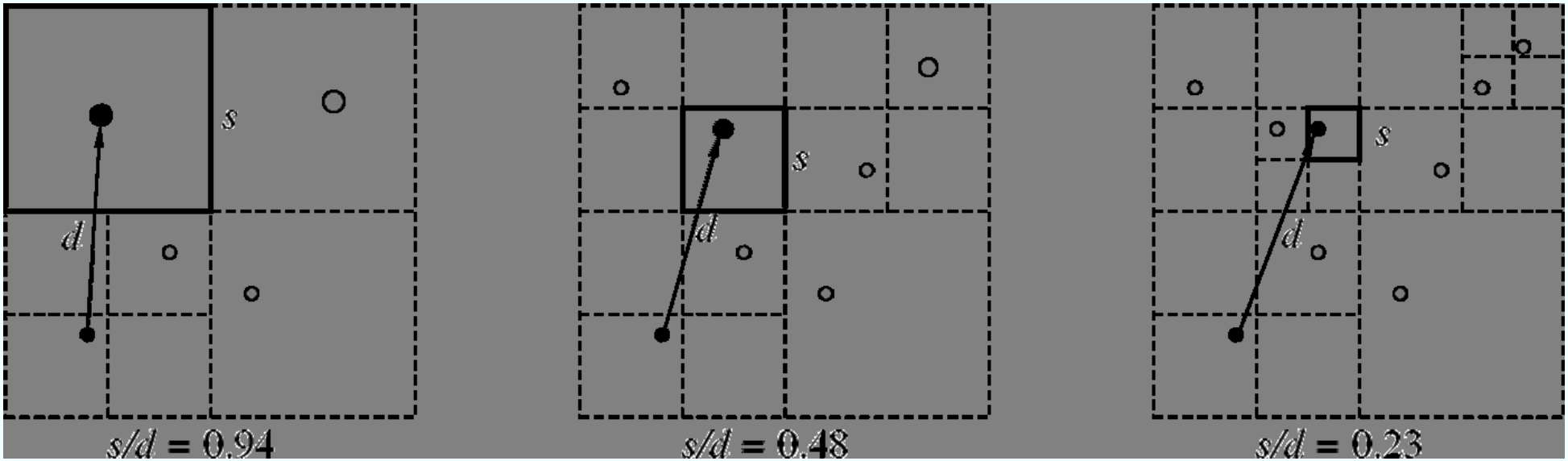


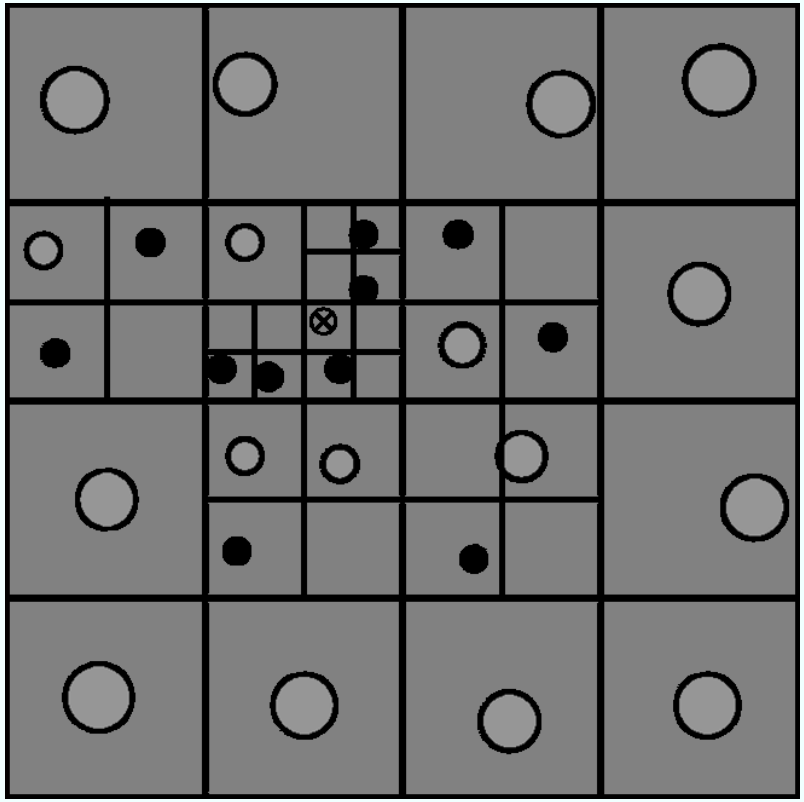


- Barnes-Hut Multipole Acceptance Criterion (MAC):

cluster included in force sum if $\dfrac{s(\text{size})}{d(\text{distance})} \leq \theta$

- Opening angle: $0.0 < \theta < 0.7$


s
d
s/d = 0.94
s
d
s/d = 0.48
s
d
s/d = 0.23



For typical values $\theta = 0.3 - 1.0$, get $N_{\text{list}} \sim \log(N)/\theta^2$ (maximum $N - 1$!)

Potential:

$$\Phi(r) = \sum_{m}^{N_{list}} \left(\frac{M^m}{r} + \frac{r_\alpha D^m_\alpha}{r^3} + \frac{1}{2} \frac{r_\alpha r_\beta Q^m_{\alpha\beta}}{r^5} + ... \right)$$

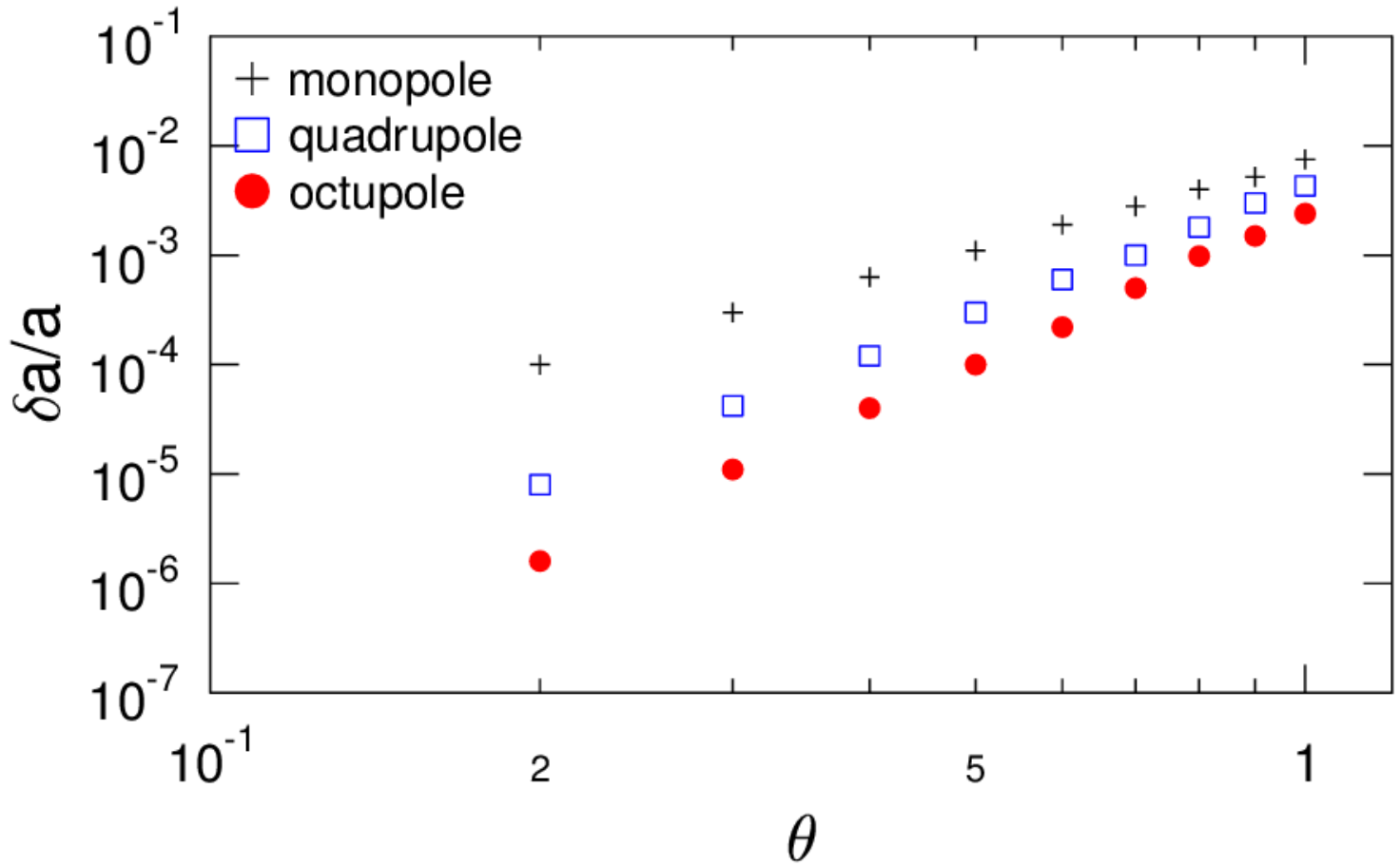
Fields:

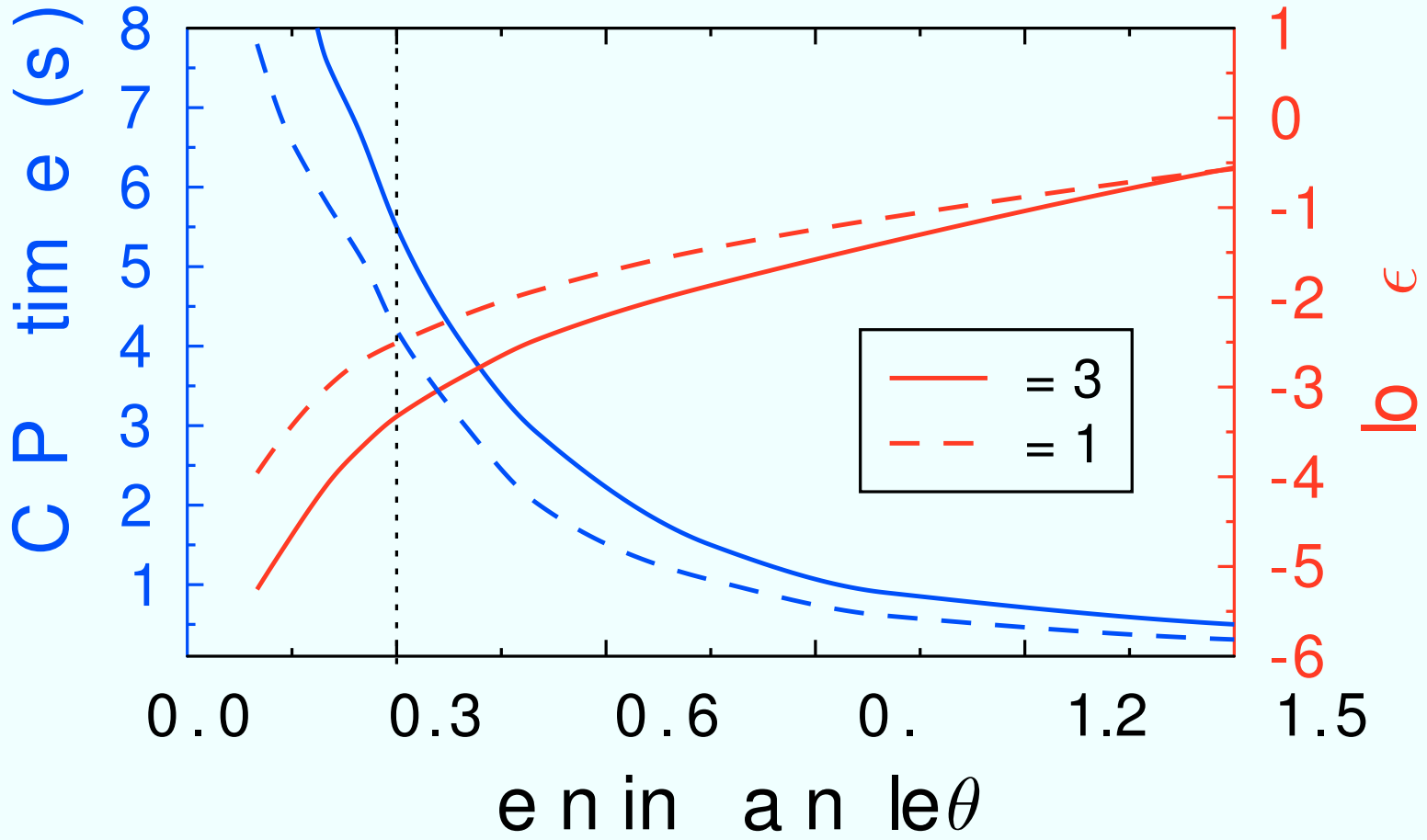
$$\boldsymbol{E}(r) = -\nabla \Phi(r),$$

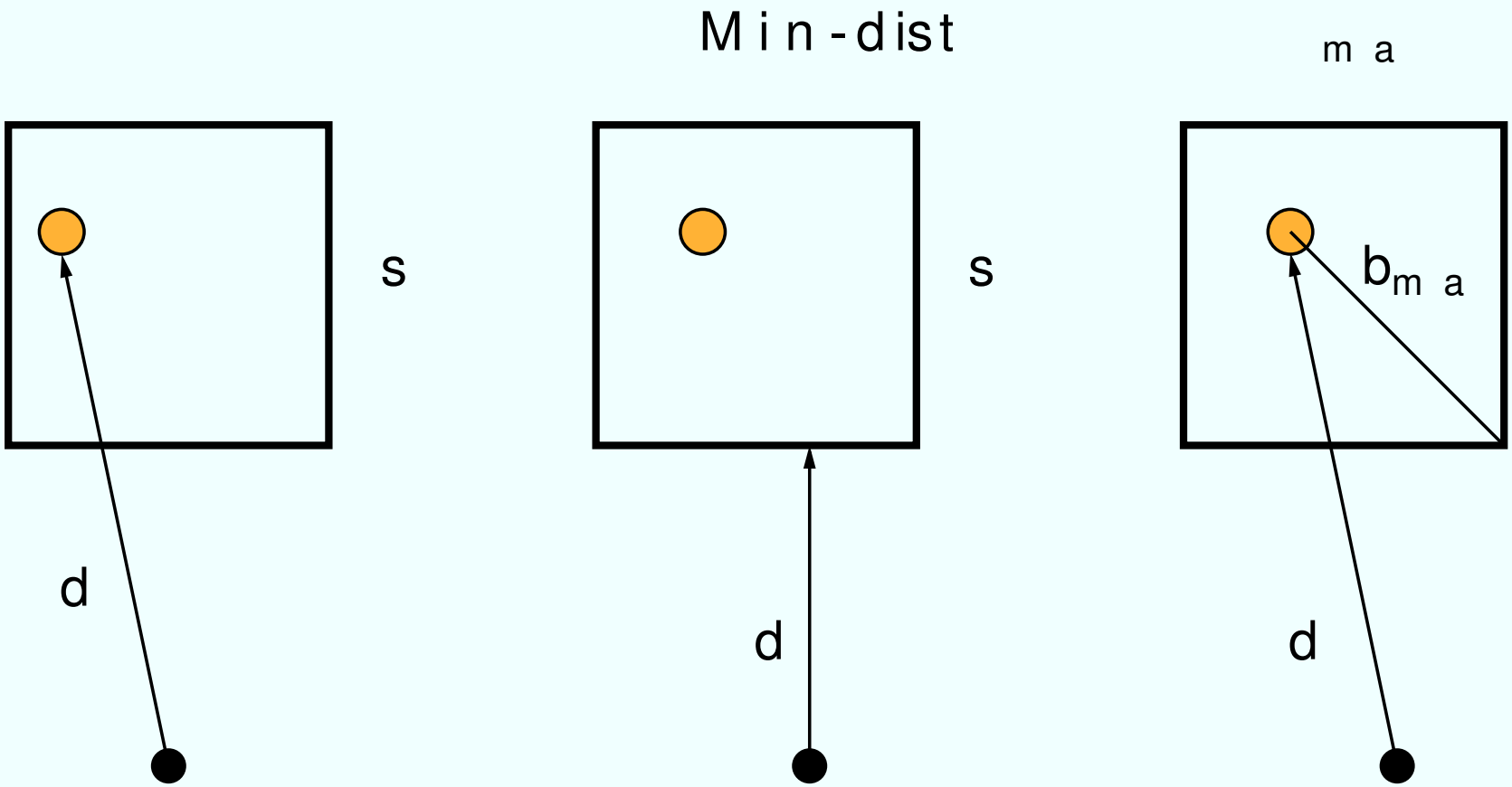
Multipole moments of nodes m:

$$\begin{aligned} M &= \sum q, \\ D_\alpha &= \sum q r_\alpha, \\ Q_{\alpha\beta} &= \sum q(3 r_\alpha r_\beta - r^2 \delta_{\alpha\beta}). \end{aligned}$$

17 – Outline


monopole
quadrupole
octupole
$\delta a/a$
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-1}
2
5
1
θ


C P tim e (s)
8
7
6
5
4
3
2
1
1
0
-1
-2
-3
-4
-5
-6
lo ε
= 3
= 1
0.0
0.3
0.6
0.
1.2
1.5
e n in a n le θ


Min-dist
m a
s
d
s
d
$b_{m\ a}$
d

Generalized algebraic kernel (Coulomb, Plummer, r^{-n}, etc.):

$$g_\lambda(r) = \sum_{l=0}^{\lambda} \frac{a_l \cdot r^{2l}}{(r^2+\sigma^2)^{\lambda+\frac{1}{2}}}$$

Plummer:

$$g_0(r) = \frac{a_0}{(r^2+\sigma^2)^{1/2}}$$

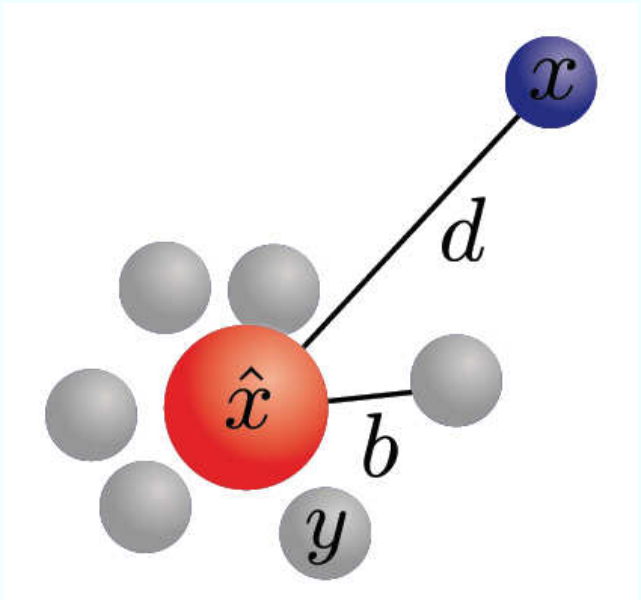
High order (vortex):

$$g_3(r) = a_3 \cdot \frac{r^2+\frac{3}{2}\sigma^2}{(r^2+\sigma^2)^{\frac{3}{2}}}$$

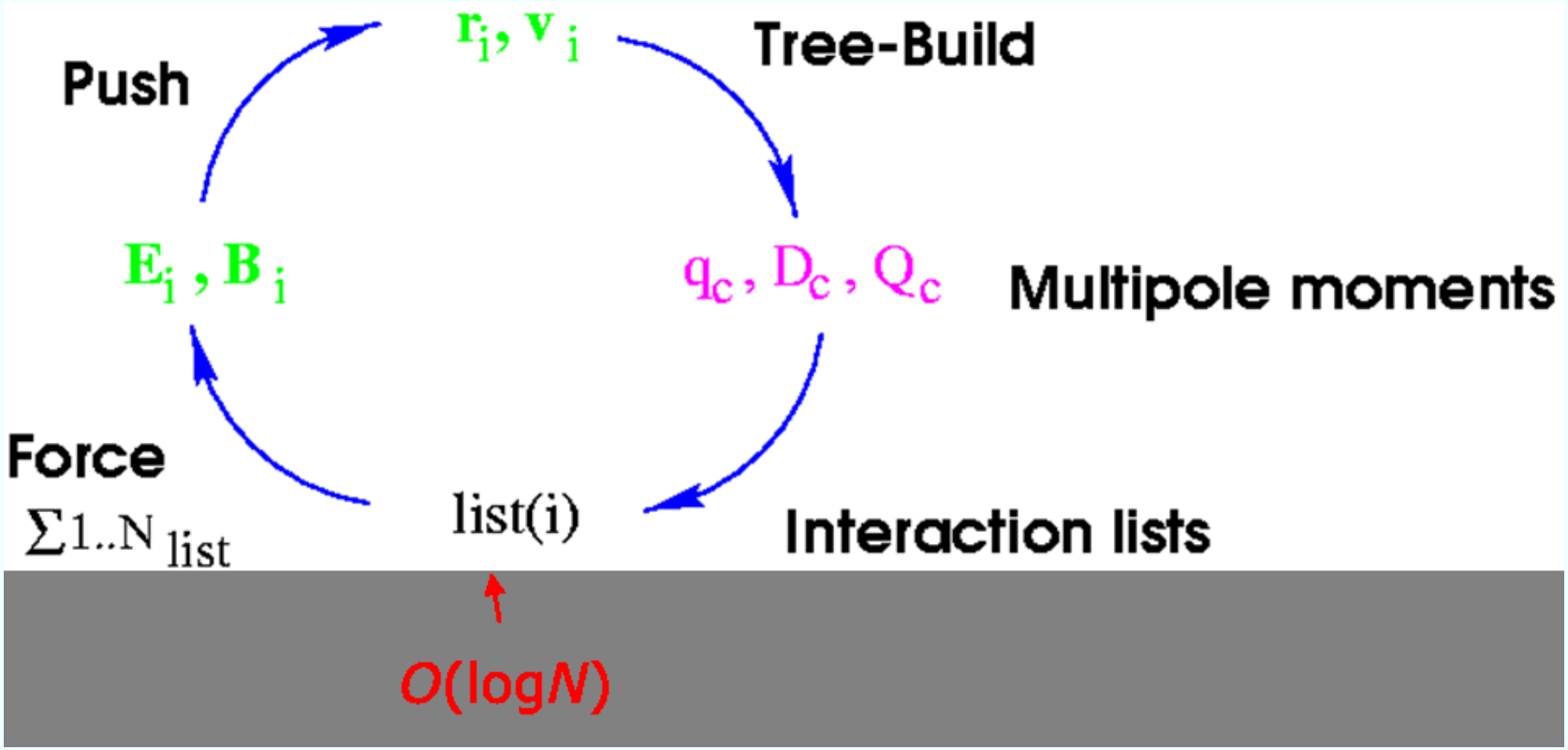
Can show that error has upper bound:

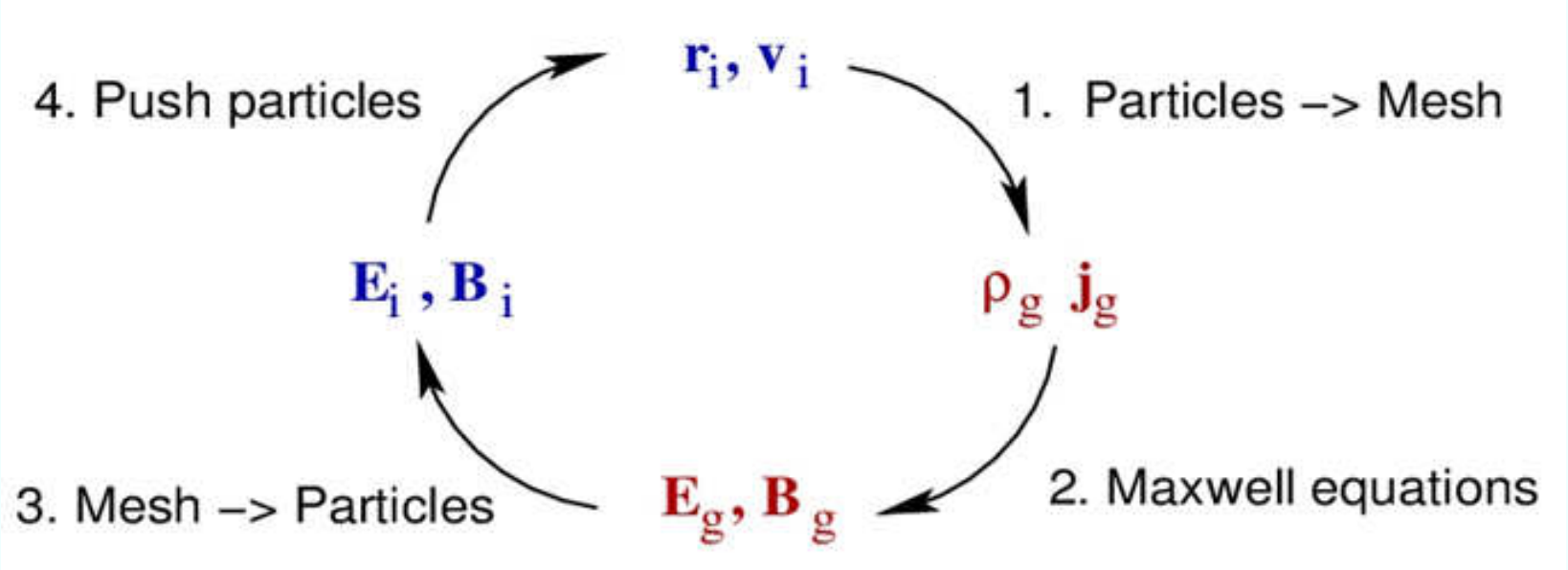
$$\varepsilon^{\tau}_{(p+1)} \leq \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{M_0}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1}$$

with monopole term M_0, $b = \sup|y - \hat{x}|$, polynomial $\mathcal{D}$ of rank $2\tau - 1$.



– See R. Speck, PhD Thesis, U. Wuppertal (2011)


r_i, v_i
Tree-Build
Push
E_i, B_i
q_c, D_c, Q_c
Multipole moments
Force
$\Sigma 1..N_{list}$
list(i)
Interaction lists
O(logN)


$\mathbf{r}_i, \mathbf{v}_i$
1. Particles –> Mesh
ρ_g $\mathbf{j}_g$
2. Maxwell equations
$\mathbf{E}_g, \mathbf{B}_g$
3. Mesh –> Particles
$\mathbf{E}_i, \mathbf{B}_i$
4. Push particles

- Download and unpack the package estree.tar.gz provided. Follow the instructions in README.compile to build the code binary. You will need a fortran compiler and make utility to do this - no problem on Linux systems - use Cygwin or MinGW for Windows. The GLE graphics package will come in handy too.
- This code comes with a large number of diagnostic routines, most of which won't be of interest - see the README.compile for a brief outline of the structure. A couple of run scripts (.sh) are provided to get you started:
  - treetest.sh
    Static configuration for performance tests and diagnosing the tree structure. The latter part is buggy, but you can use `gle tree-box-1.gle` to obtain a 2D tree.
  - plasma-eqm.sh
    Sets up a thermal electron-ion plasma. Can be used for energy/momentum conservation tests and adapted to simulate more complex plasma geometries.
- Using the treetest.sh script investigate the performance of the tree algorithm compared to direct summation (routines found in src/enerpp.f, src/forcpp.f, src/errorest.f). Here you can start by varying the multipole order `multi`, opening angle (MAC) `theta` and particle number `ne, ni`. Compare your results with the error and computation time scalings from the lecture notes. For how many particles N does the tree code start to perform better (break-even point)?

Further possible investigations (choose 1):

- Extend the performance study to compare homogeneous vs inhomogeneous configurations with different MACs - see the do-while loop in routine intlist.f. You can replace the routines called in ranpart.f to create more interesting setups.
- Use the plasma-eqm script to investigate energy and momentum conservation. Main parameters to consider are `eps`, `theta`, `dt`. Devise a scheme – perhaps using the tree structure – to identify particles undergoing close encounters and thus requiring smaller timesteps.
- Implement octupole order corrections to force sum (or more general high-order multipole expansion). You'll need to modify the core routines charges.f, force.f, energy.f, es.h for this.

- S. Pfalzner, P. Gibbon, Many Body Tree Methods in Physics, Cambridge University Press, New York (September 2005), ISBN 0-521-01916-8.
- J. Barnes & P. Hut, *A hierarchical O(N log N) force-calculation algorithm*
- L. Hernquist, *Performance characteristics of tree codes*, Ap. J. Supp. **64**, 715 (1987).
- J. Salmon & M. Warren, *Skeletons from the treecode closet*, J. Comp. Phys. **111**,136 (1994)
- M. Winkel, R. Speck, H. Hübner, L. Arnold, R. Krause, P. Gibbon, A massively parallel, multi-disciplinary Barnes-Hut tree code for extreme-scale N-body simulations, Comp. Phys. Commun. **183**, 880 (2012).

Part V

Multipole methods 2: Fast Multipole Method

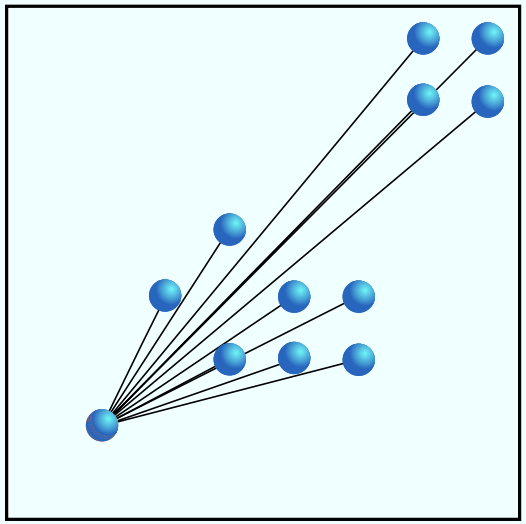
19 Fast multipole method

20 Basic FMM algorithm

21 Performance

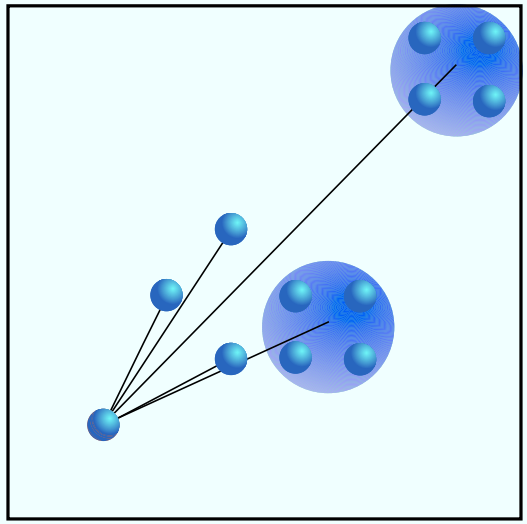
19 – Outline

Direct Summation



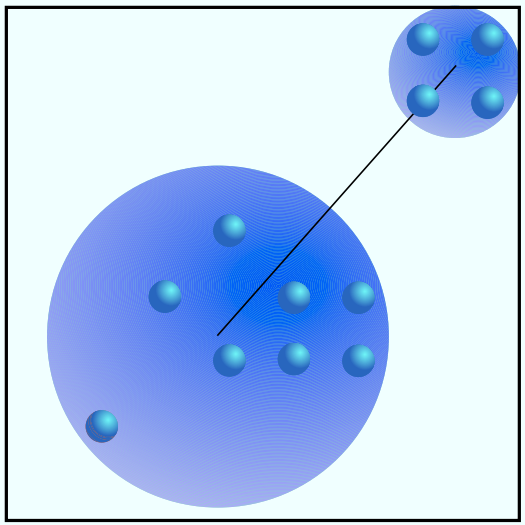
interactions: N^2

Treecode



$N \log N$

Fast Multipole Method

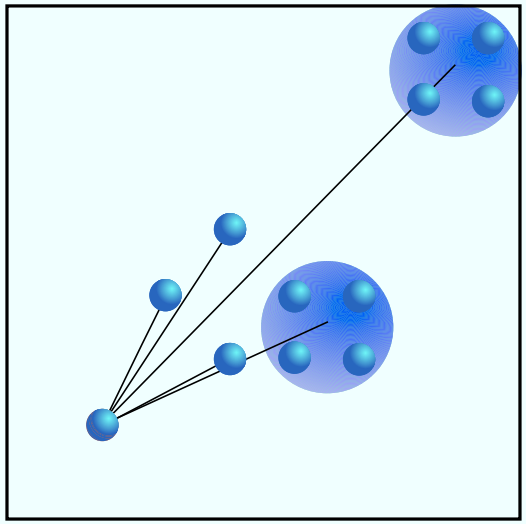


N

Barnes-Hut Tree Code
particle-cluster interactions $\rightarrow O(N \log N)$
J. Barnes & P. Hut, Nature (1986)

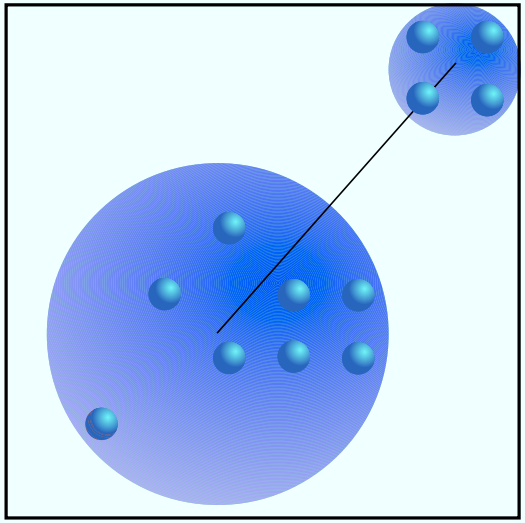
Fast-Multipole Method
cluster-cluster interactions $\rightarrow O(N)$
L. Greengard & V. Rohklin, J. Comp. Phys. (1987)

Treecode



- $O(N \log N)$
- dynamic load-balancing
- versatility + adaptability
- easy exchange of interaction kernel

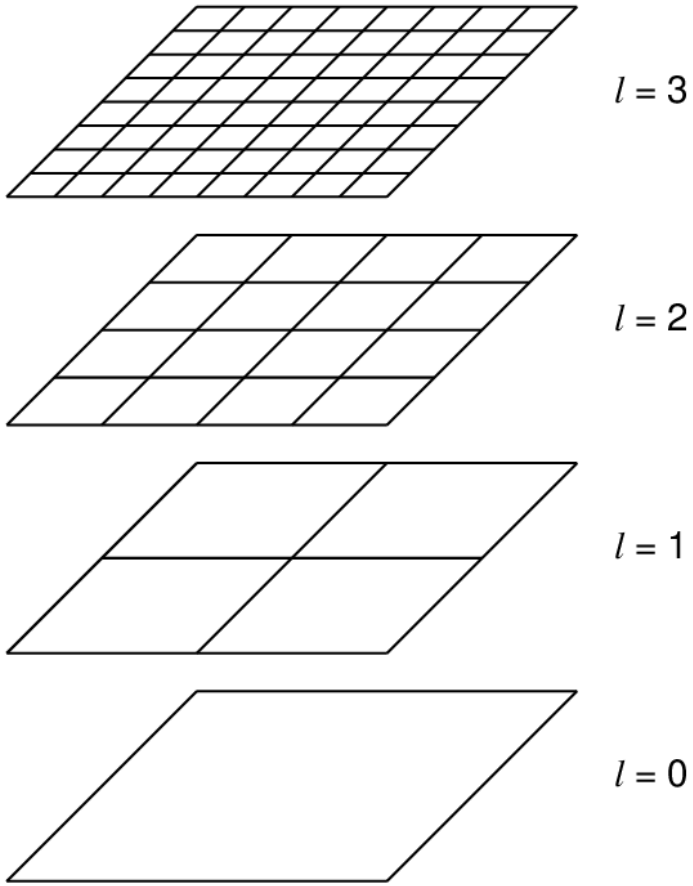
FMM



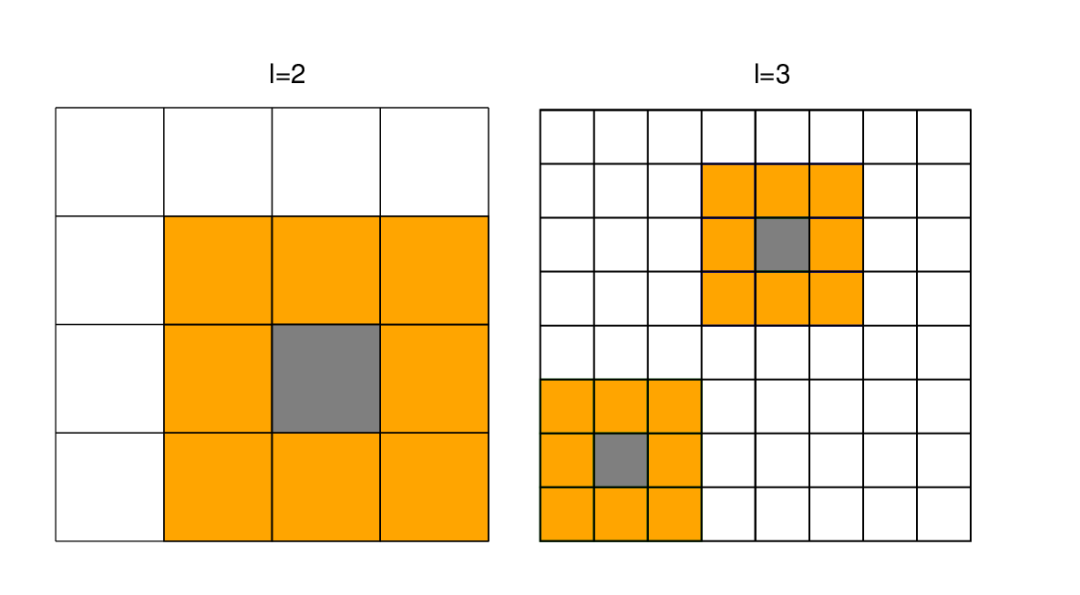
- $O(N)$
- rigorous error control
- high (machine) accuracy
- fast!

20 – Outline

- FMM makes rigorous use of the fact that a multipole expansion to infinite order contains the total information of a particle distribution.
- Like BH tree-code, interaction between near-neighbours is calculated by direct particle–particle force summation, and more distant particles are treated separately.
- But in FMM the distant region is treated as a *single* 'far-field' contribution, calculated by a high-order multipole expansion.
- By carefully choosing boundary between near- and far-field, can reduced complexity to $O(N)$.



- Root box contains all particles
- Tree refined up to max. refinement level L
- For uniform distribution, this implies $L = \log_8 N$



The FMM tree structure is used to build ‘near-neighbour’ lists of boxes at each refinement level l.
Near-neighbours are defined as the box itself and any box at the same level with which it shares a boundary point.

By contrast, a box on the same level which is *not* in a near-neighbour list is well separated

Using the usual conventions (see Jackson) for spherical harmonics , the generalized version of (59) is:

$$\Phi(\mathbf{r}) = 4\pi \sum_{l,m} \frac{M_{lm} Y_{lm}(\theta, \phi)}{(2l+1) r^{l+1}} \tag{67}$$

in spherical coordinates (r, θ, ϕ) relative to an origin O, with multipole moments given by:

$$M_{lm} = \sum_i q_i r_i^l Y_{lm}^*(\theta_i, \phi_i), \tag{68}$$

where the charges now have the coordinates (r_i, θ_i, ϕ_i).

The charges are assumed to have positions s_i, such that $z_i = s\cos\theta$, $x = s\sin\theta\cos\phi$, and $y = s\sin\theta\sin\phi$. The first few spherical harmonics up to 2nd order (ie: $p = 2$) are as follows:

$$\begin{aligned}
Y_{00} &= \frac{1}{\sqrt{4\pi}}, \\
Y_{10} &= \sqrt{\frac{3}{4\pi}}\cos\theta, \\
Y_{11} &= -\sqrt{\frac{3}{8\pi}}\sin\theta e^{i\phi}, \\
Y_{20} &= \sqrt{\frac{5}{4\pi}}\left(\frac{3}{2}\cos^2\theta - \frac{1}{2}\right), \\
Y_{21} &= -\sqrt{\frac{15}{8\pi}}\sin\theta\cos\theta e^{i\phi}, \\
Y_{22} &= \frac{1}{4}\sqrt{\frac{15}{2\pi}}\sin^2\theta e^{2i\phi}.
\end{aligned} \tag{69}$$

Further terms can be found from:

$$Y_{lm}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi}$$

where

$$P_l^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^{m+l}}{dx^{m+l}} (x^2-1)^l$$

are the Legendre polynomials. The expressions in (**??**) are related to the cartesian multipole moments as follows:

$$\begin{aligned}
q_{00} &= \frac{Q}{\sqrt{4\pi}} & q_{20} &= \frac{1}{2}\sqrt{\frac{5}{4\pi}} Q_{zz} \\
q_{10} &= \sqrt{\frac{3}{4\pi}} D_z & q_{21} &= -\frac{1}{3}\sqrt{\frac{15}{8\pi}} (Q_{xz} - iQ_{yz}) \\
q_{11} &= -\sqrt{\frac{3}{8\pi}} (D_x - iD_y) & q_{22} &= \frac{1}{12}\sqrt{\frac{15}{2\pi}} (Q_{xx} - Q_{yy} - 2iQ_{xy})
\end{aligned}$$

$$\begin{aligned}
\Phi_0 &= 4\pi q_{00}\frac{Y_{00}}{r} = \frac{Q}{r} \\
\Phi_1 &= \sum_{m=-1}^{1}\frac{4\pi}{3}q_{1m}\frac{Y_{1m}}{r^2} = \sum_j \frac{r_j D_j}{r^3} \\
\Phi_2 &= \sum_{jk}\frac{1}{2}Q_{jk}\frac{r_j r_k}{r^5}
\end{aligned} \tag{70}$$

where

$$\begin{aligned}
Q &= \sum q \\
D_j &= \sum q s_j \\
Q_{jk} &= \sum q(3s_j s_k - s^2\delta_{jk})
\end{aligned}$$

are the monopole, dipole and quadrupole moments respectively.

Multipole moments M_{lm} are first computed for each box at the highest refinement level L, but this time relative to the *centre of the box* rather than the centre of charge of the particles. The maximum number of terms p in the multipole expansion is chosen such that**??**:

$$(\sum_{i}^{N_{box}} \mid q_i \mid)2^{-p} \leq \epsilon, \tag{71}$$

where ϵ is the desired precision.

Next, the multipole moments on the next coarsest refinement level $l = L - 1$ are calculated, and just as for the tree method, the shifted multipole moments of the child cells can be used to obtain the moments of the parent cell. Shifting the origin O to O' by a translation vector $\mathbf{r}_t$, a transformation of the multipole moments is needed to obtain the expansion in terms of the new vector $\mathbf{r}' = \mathbf{r} - \mathbf{r_t}$ relative to O'. This transformation is given by:

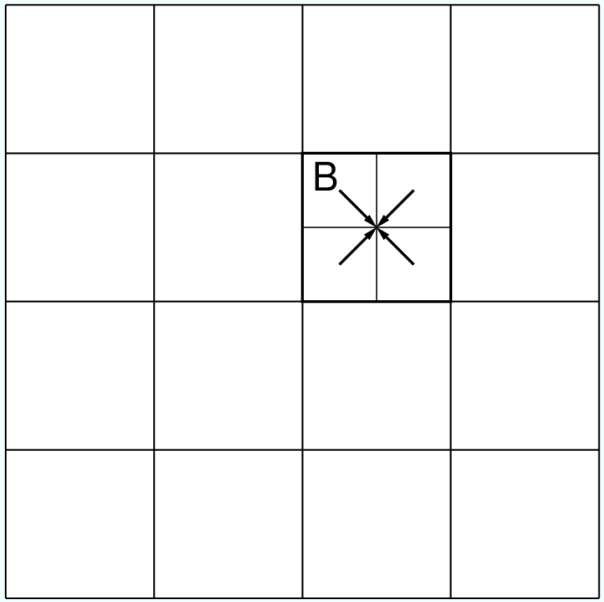
$$M'_{l'm'} = \sum_{l,m} T^{MM}_{l'm',lm} M_{lm}, \tag{72}$$

with the transformation matrix

$$T^{MM}_{l'm',lm} = 4\pi \frac{(-r_t)^{l'-1} Y^*_{l'-l,m'-m}(\theta_t, \phi_t) a'_{l'-l,m'-m} a_{lm} (2l'+1)}{2(l+1)[2(l'-l)+1] a_{l'm'}},$$

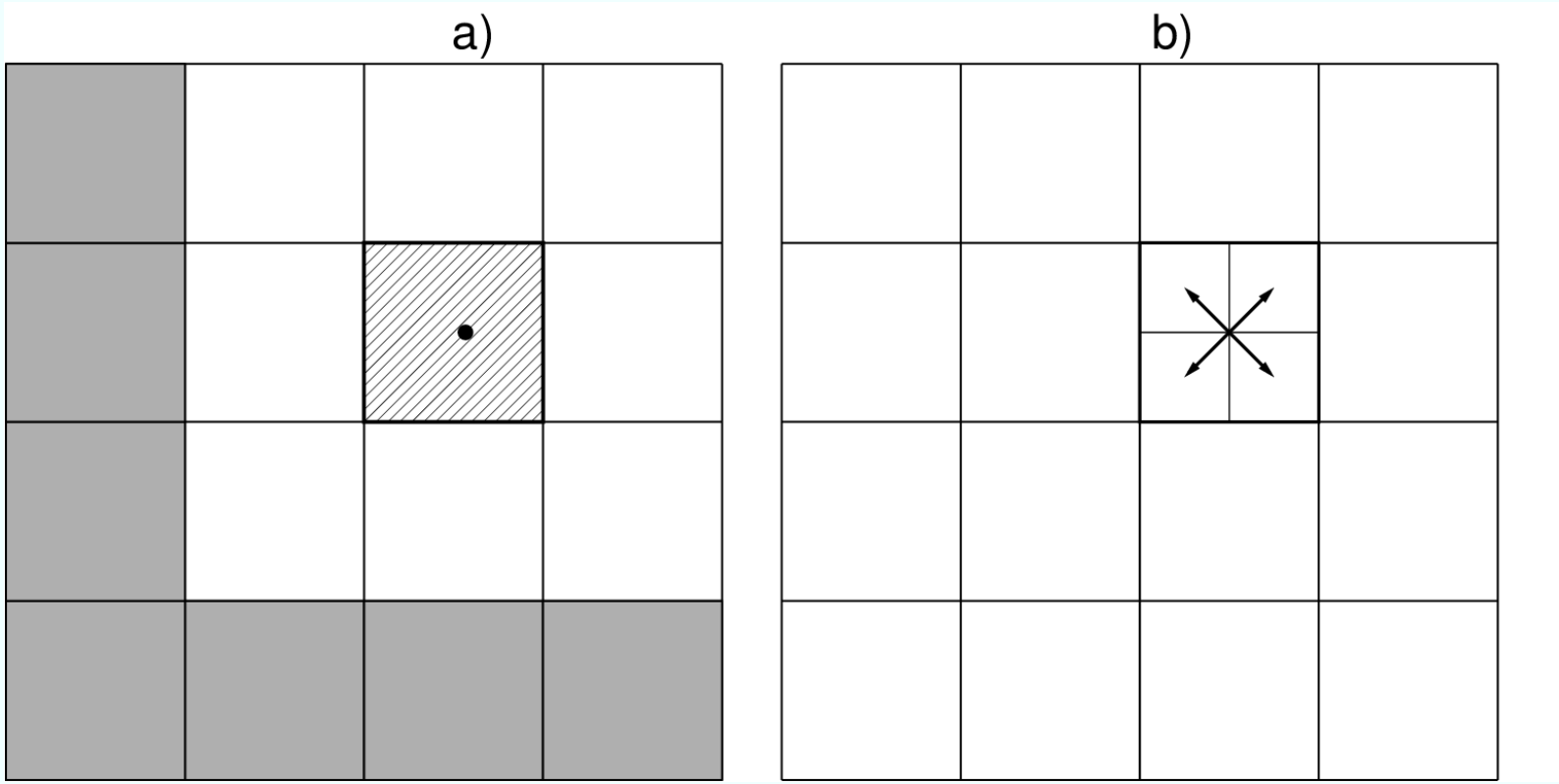
and a_{lm} is defined as

$$a_{lm} = (-1)^{l+m} \frac{2(l+1)^{1/2}}{[4n(l+m)!(l-m)!]^{1/2}}.$$



Eq. 72 is just a generalization of the shifting relations used for the (quadrupole order) tree code, with the main difference that the moments are calculated relative to the centre of the cell (see Fig. 2). In this fashion, the moment expansions are carried up to to root level $l = 0$, which ultimately contains an p-term multipole expansion of the whole system.

Next, a downward pass (from $l = 0$ to L) is made, in which a careful distinction is made in the interaction lists for the boxes.
Actually *three* regions: the near field, the interactive field and the far field.
The near field consists of the neighbouring cells; the far field is the entire simulation box *excluding* the cell in question and its neighbours. The interactive field is the part of the far field that is contained in the near field of this cell's *parents*.



a) List of well-separated boxes from far-field which contribute to
b) Local expansion of the hatched box in at level $l = 2$

In this downward pass, each multipole expansion is converted into a *local expansion*, i.e. a Taylor expansion of the potential about the centres of all well-separated boxes at each level. Using the notation above, this can be expressed thus:

$$\Psi(\mathbf{r}) = 4\pi \sum_{l,m} L_{lm} r^l Y_{lm}(\theta, \phi), \tag{73}$$

where L_{lm} are referred to as the local moments of the Taylor series expansion, obtained from the original multipole moments by the transformation:

$$L_{l'm'} = \sum_{l,m} T^{LM}_{l'm',lm} M_{lm}, \tag{74}$$

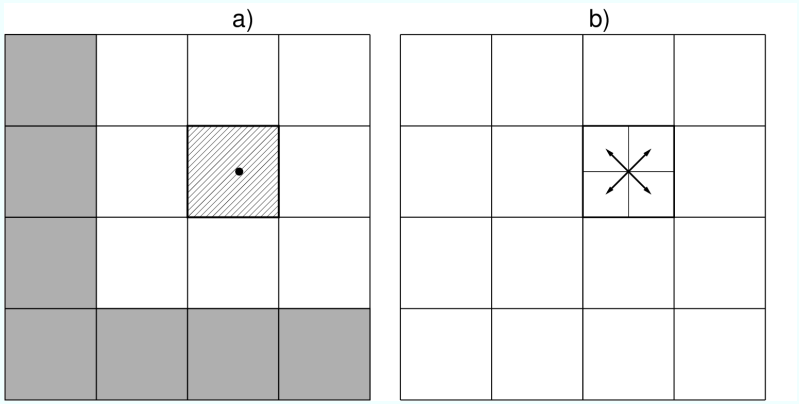
and the transformation matrix $T^{LM}_{l'm',lm}$ is now

$$T^{LM}_{l'm',lm} = 4\pi \frac{(-1)^{l+m} Y^*_{l'+l,m'-m}(\theta_t, \phi_t) a_{l,m} a_{l'm'}}{r_t^{l'+l+1}(2l+1)(2l'+l) a_{l'+l,m'-m}}. \tag{75}$$

Local expansion validity consists of all boxes at the same level which are not near neighbours. At levels $l = 0$ and $l = 1$, there are no boxes which fulfill this requirement, so we just set

$$\Psi_0 = \Psi_1 = 0$$

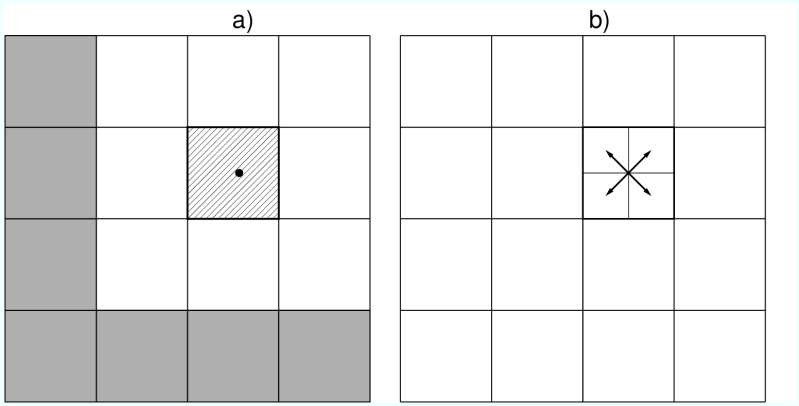
Levels $l = 2$ to $l = L$: for each box on level l, the local expansion of the parent box is shifted to the centre of each of its child boxes as in b):



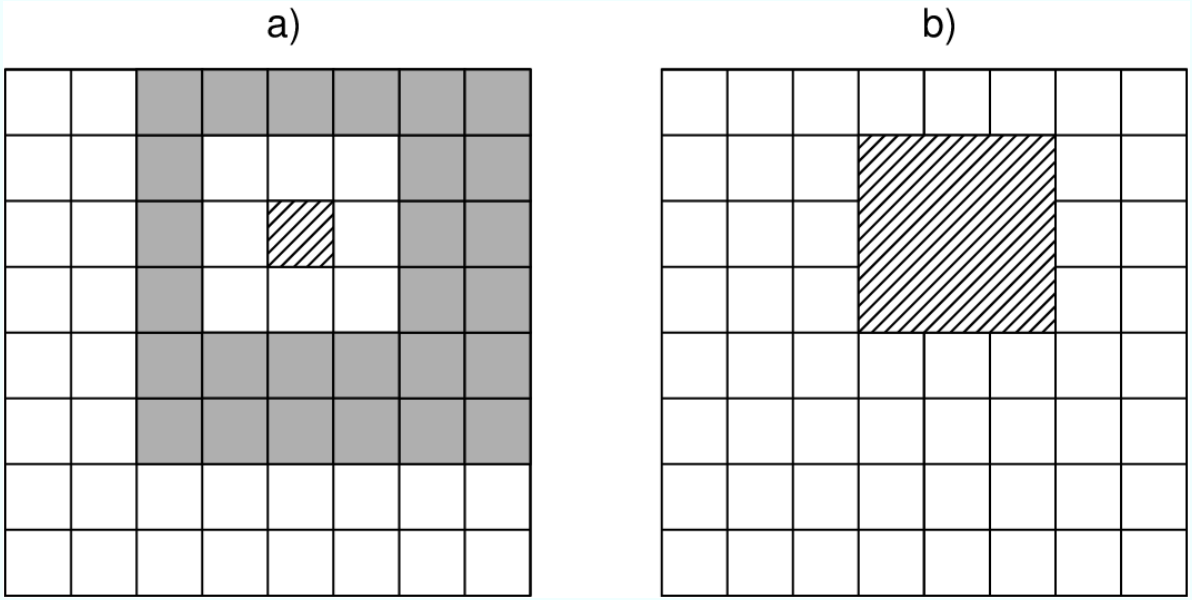
$$L'_{l'm'} = \sum_{l,m} T^{LL}_{l'm',lm} L_{lm},$$

with

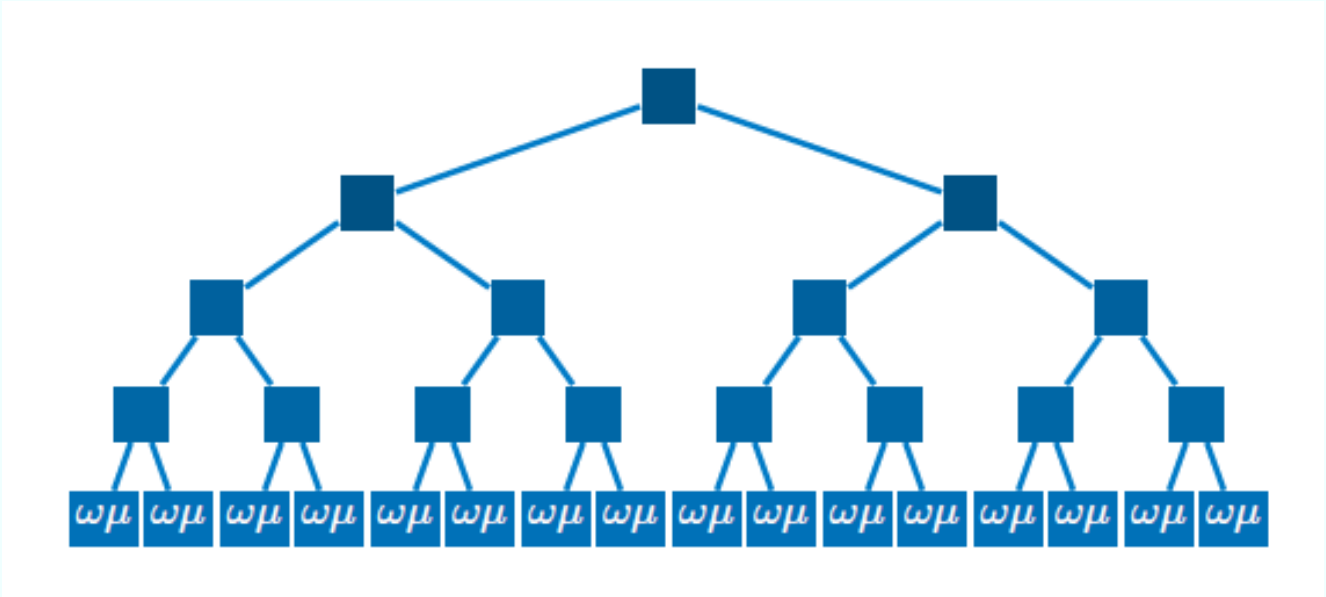
$$T^{LL}_{l'm',lm} = 4\pi \frac{r_t^{l-l'} Y_{l'-l,m'-m}(\theta_t, \phi_t) a_{l'm'} a_{l-l',m-m'}}{(2l'+1)[2(l-l')+1] a_{lm}}.$$



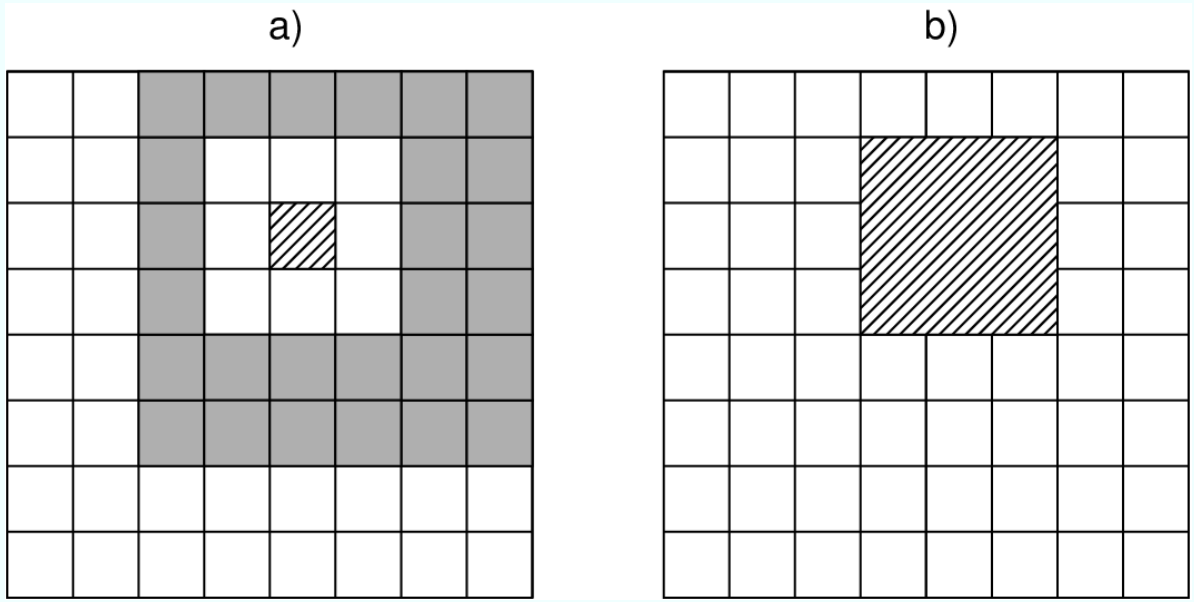
In this local expansion of the child cells there are contributions missing. These are the boxes that do not touch the current cell, but do not contribute to the local expansion of the parent cell either – in other words, the boxes of the *interactive* field – a).



20 – Pass 4: Transfer far-field potential to particles

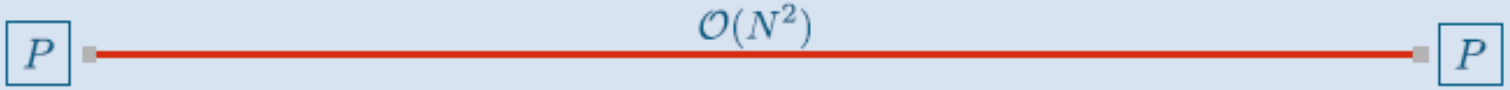


Finally, the remaining interactions with the particles in neighbouring boxes and the box itself are added by direct summation of the particle–particle interactions – b).

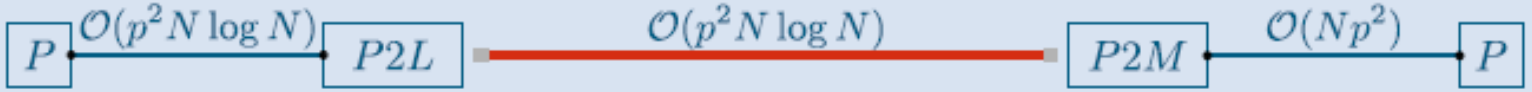


21 – Outline

Direct Summation $\mathcal{O}(N^2)$

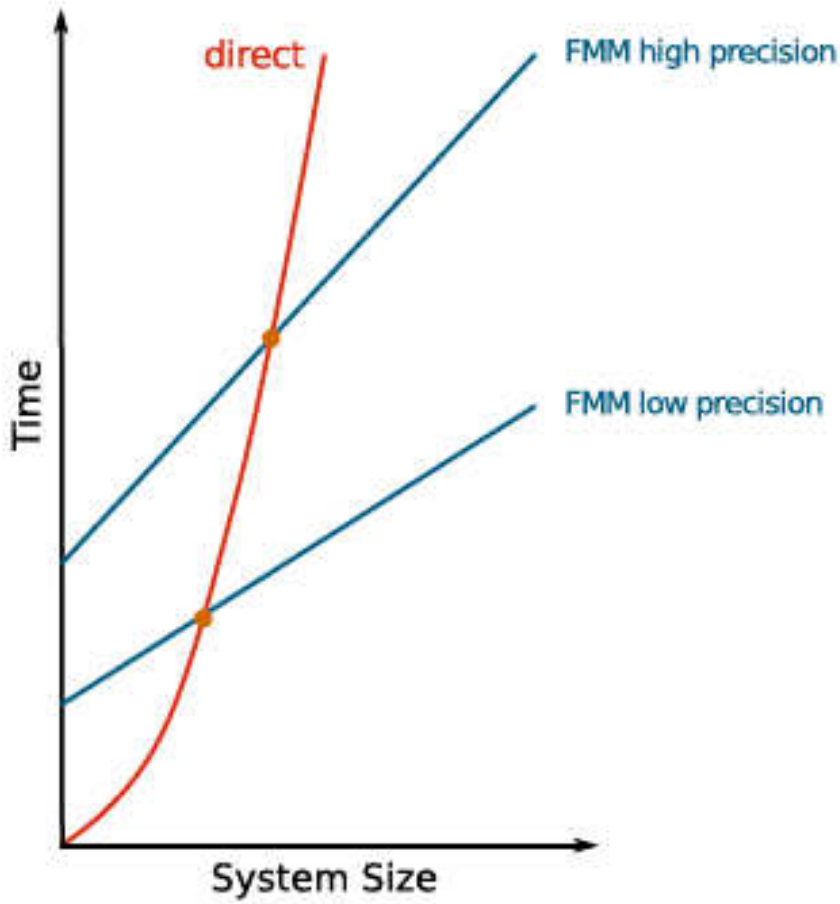


Barnes-Hut Treecode $\mathcal{O}(p^2 N \log N)$ $(p \ll N)$



Fast Multipole Method $\mathcal{O}(p^3 N)$ $(p \ll N)$

Courtesy, Ivo Kabadshow



High Precision Cross-Over

- $\Delta E = 10^{-12}$
- 4000 particles

Low Precision Cross-Over

- $\Delta E = 10^{-3}$
- 500 particles

- Carrier, Greengard, Rokhlin (1988): Adaptive FMM for non-uniform particle distributions.
- White & Head-Gordon (1994): Factorise $1/\mid r-a\mid$ expansion in terms of scaled Legendre polynomials. Compact and computationally efficient.
- Dachsel (2006): Error controlled FMM - a priori, user-defined precision.

- L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys. **73**, 325-348 (1987); reprinted in special issue: J. Comp. Phys. **135**, 280-292 (1997)
- K. E. Schmidt and M. A. Lee, J. Stat. Phys. **63**, 1223 (1991)
- C. A. White and M. Head-Gordon, *Derivation and efficient implementation of the fast multipole method*, J. Chem. Phys. **101**, 6593–6605 (1994) - recommended !
- I. Kabadshow, *The Fast Multipole Method: a linear scaling Coulomb solver*, Lecture notes from Heraeus Summer School, Jülich, Sept. 2010 (on Toledo).
- S. Pfalzner, P. Gibbon, Many Body Tree Methods in Physics, Cambridge University Press, New York (September 2005), ISBN 0-521-01916-8.
- J. D. Jackson, Classical Electrodynamics, Wiley (1975)

Part VI

Parallel Methods

Part VI

Parallel Methods

22 – Outline

- Computation as replacement/reinforcement of real experiments, which may be too large/small, complex, expensive or impossible.
- Examples: climate, astrophysics, nano-materials, aerodynamics
- Realistic simulations need huge computational resources.
- Problem size – memory and time – quickly becomes too large for single processor.

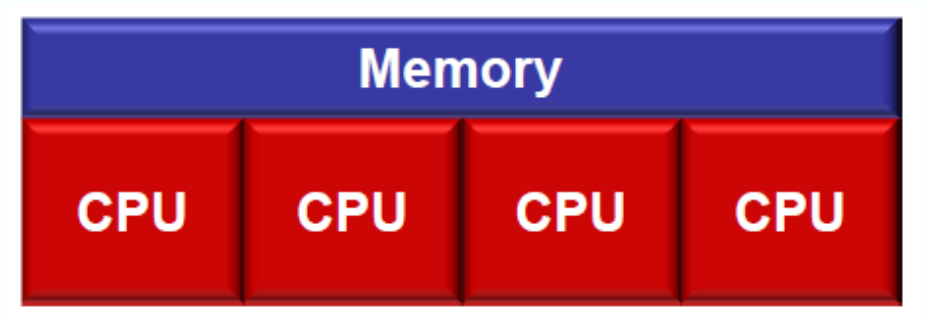
- Parallel Computing: solving task by simultaneous use of multiple processors within a single unified architecture
- Distributed Computing: solving task by simultaneous use of multiple processors of isolated, heterogeneous computers
- Embarrassingly Parallel: solving many similar, but independent, tasks; e.g., parameter sweeps (farming)
- Supercomputing: Use of the fastest and biggest machines to solve large problem
- High Performance Computing (HPC): Solving a problem via supercomputers + fast networks + large storage + visualization

23 – Outline

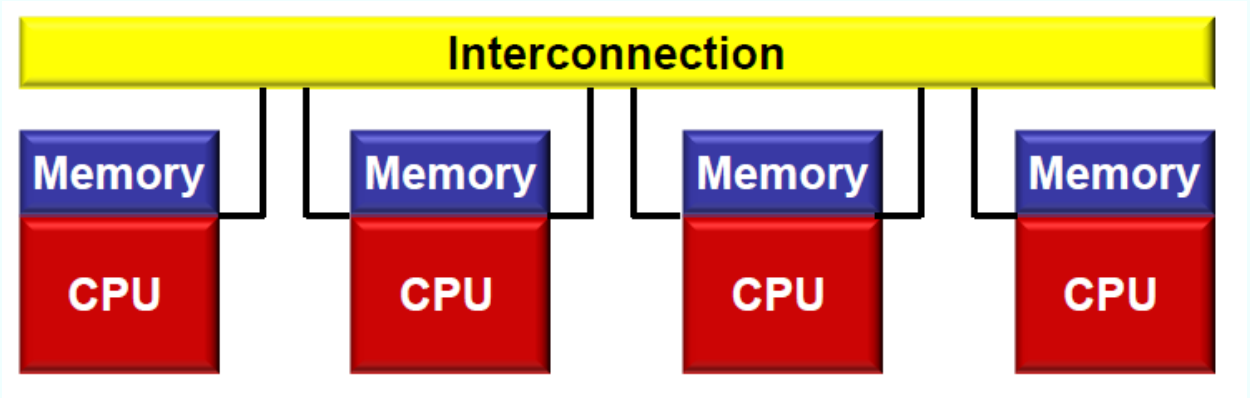
General classification:

	Single Instruction	Multiple Instructions
Single Data	SISD	MISD
Multiple data	SIMD	MIMD

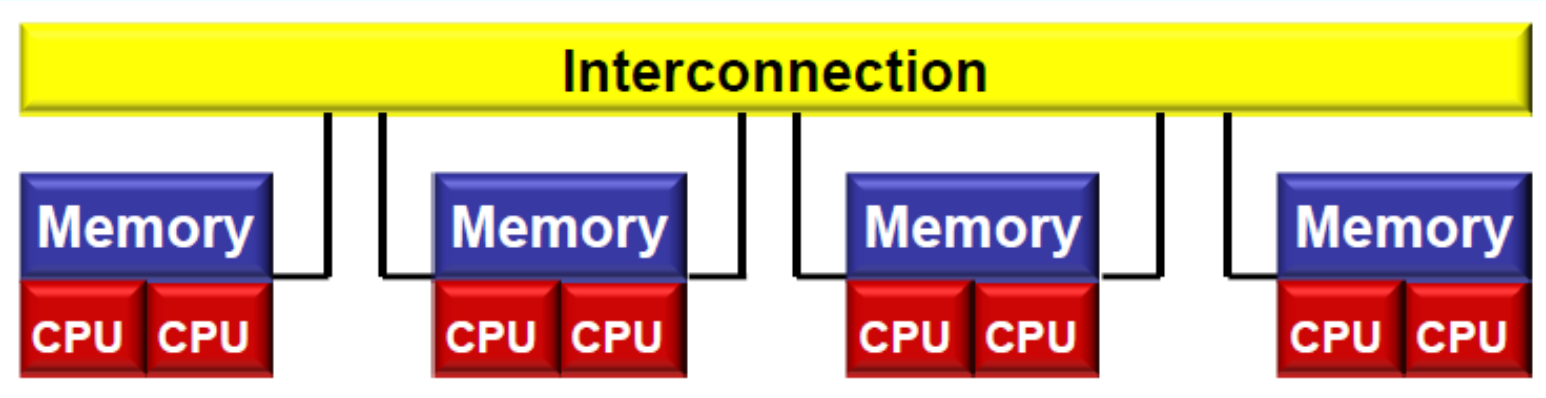
- SISD: Uniprocessor, Pentium
- SIMD: SSE instruction of x86, vector processors
- MISD: rare, used for fault tolerance
- MIMD: Multiprocessor architecture - basis of almost all HPC systems



- All CPUs share the same memory
- Single address space with either:
  - Uniform memory access (UMA), eg: Symmetric MultiProcessor (SMP)
  - Non-uniform memory access (NUMA), eg: two (or more) SMP linked together

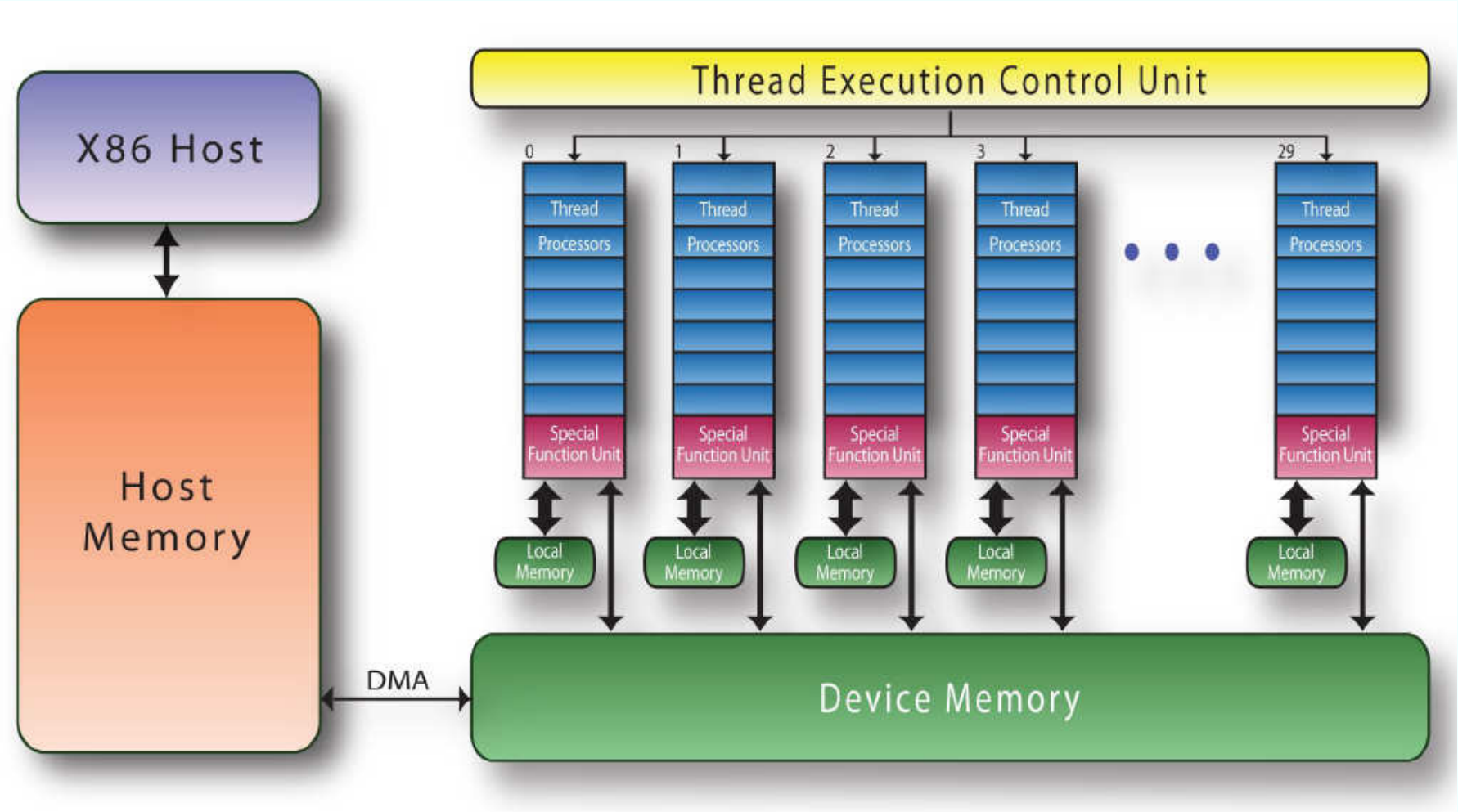


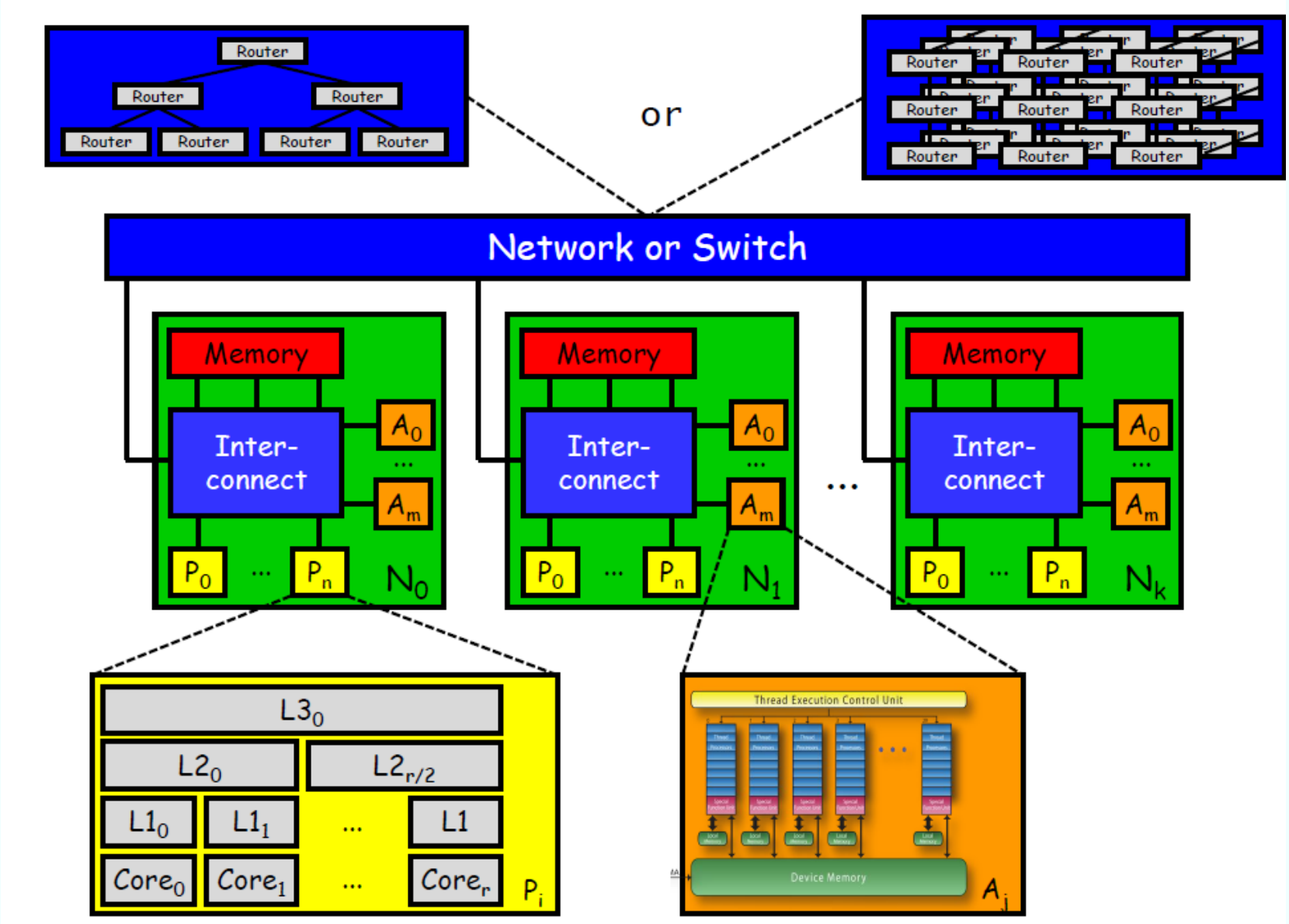
- Each CPU has its own memory and address space
- Data exchange between memory of different CPUs
  - Via interconnect
  - Explicit data transfer necessary (message passing)



- SMP with up to 16 cores (UMA within each SMP)
- Several SMP are combined in one compute node (CN) (NUMA between the SMP)
- CN are connected via a network
- Special nodes: CN, I/O nodes, login nodes

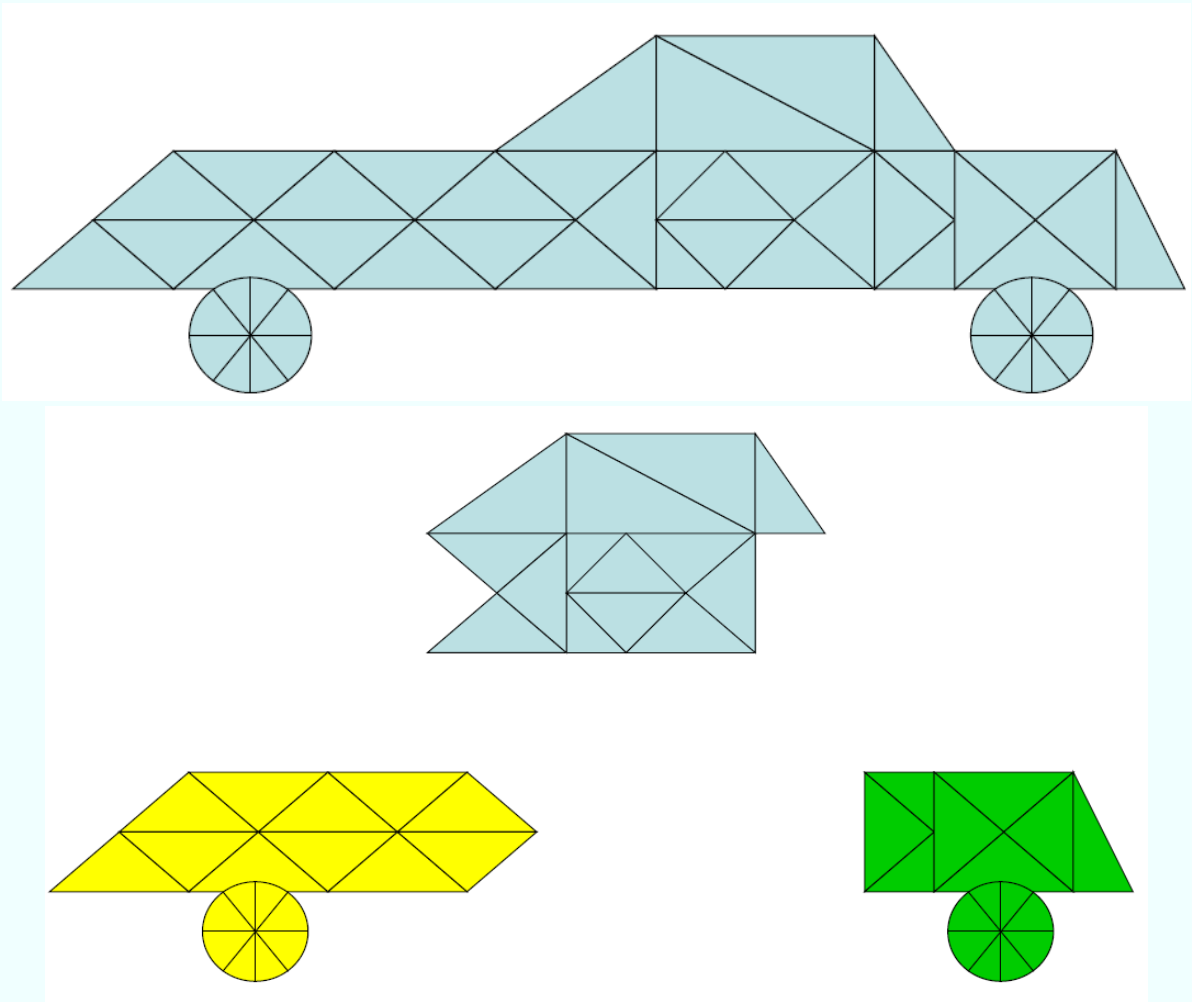
- Special hardware for accelerating floating-point intensive code
- General Purpose computing on Graphics Processing Units (GPGPU)
- Intel Many Integrated Core (MIC) Architecture



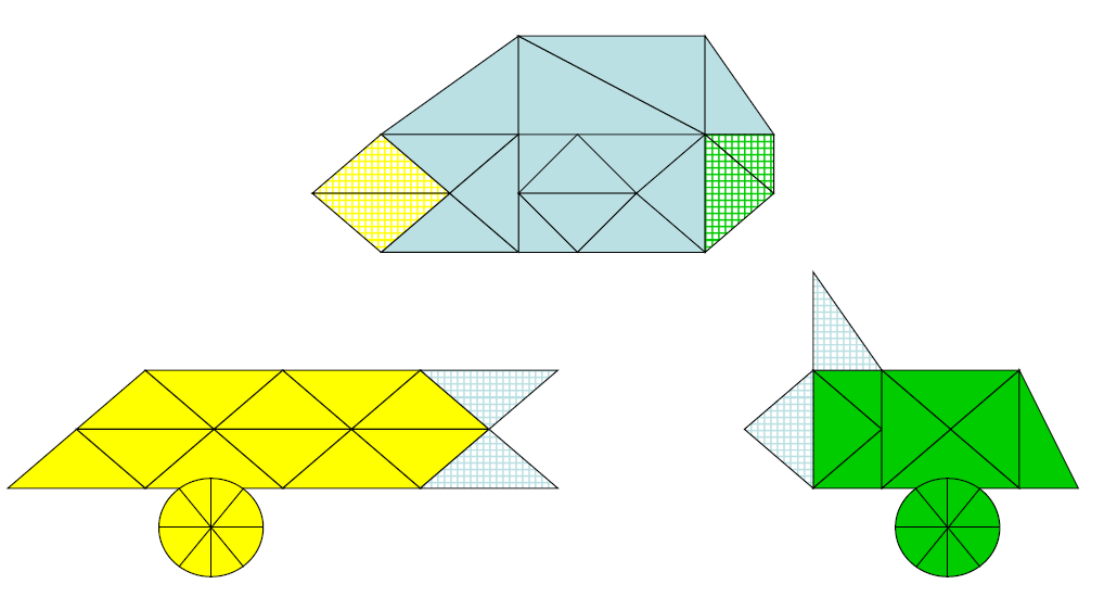

Router
Router
Router
Router
Router
Router
Router
or
Router
Router
Router
Router
Router
Router
Router
Router
Router
Network or Switch
Memory
Inter-connect
A_0
...
A_m
P_0
...
P_n
N_0
Memory
Inter-connect
A_0
...
A_m
P_0
...
P_n
N_1
...
Memory
Inter-connect
A_0
...
A_m
P_0
...
P_n
N_k
$L3_0$
$L2_0$
$L2_{r/2}$
$L1_0$
$L1_1$
...
L1
$Core_0$
$Core_1$
...
$Core_r$
P_i
Thread Execution Control Unit
...
Device Memory
A_j

24 – Outline

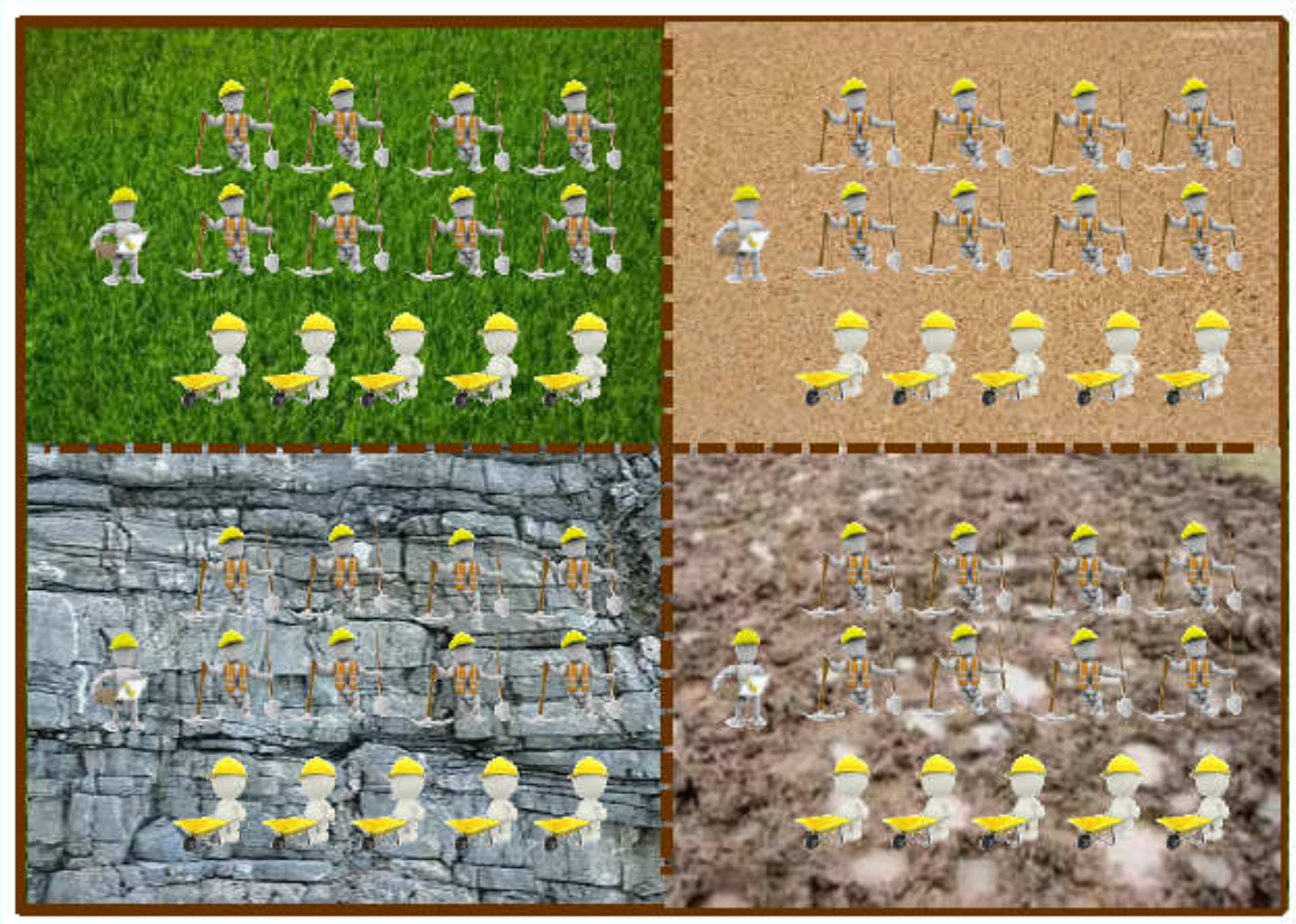
- Goal: Divide work and/or communication between processors
- Two main approaches:
  - Domain decomposition: partition real or conceptual space (eg coordinates)
  - Functional decomposition: different processors work on different types of task
- Functional decomp. rarely scales to many processors, so most programs parallelized using some form of domain decomposition.



- Processors need to keep track of information which might be required from or needed by neighbours in other processors
- Use ghost cells (halo) to copy remote neighbours, add transition table to keep track of their location and which local elements are copied elsewhere



- Goal: aim to divide work equally between processors
- Problem: work distribution not known a priori and can change dynamically



- domain decomposition:
  - convert local particle coordinates to keys
  - perform global key-sort
- build tree:
  - construct local nodes
  - define global branches
  - fill in top-level nodes
  - compute multipole properties
- tree-walk:
  - get interaction lists & compute forces for each particle

Consider coordinate $r = (2.5, 3.5, 4.0)$ with box length $L = 8.0, n_{\text{level}} = 4$.

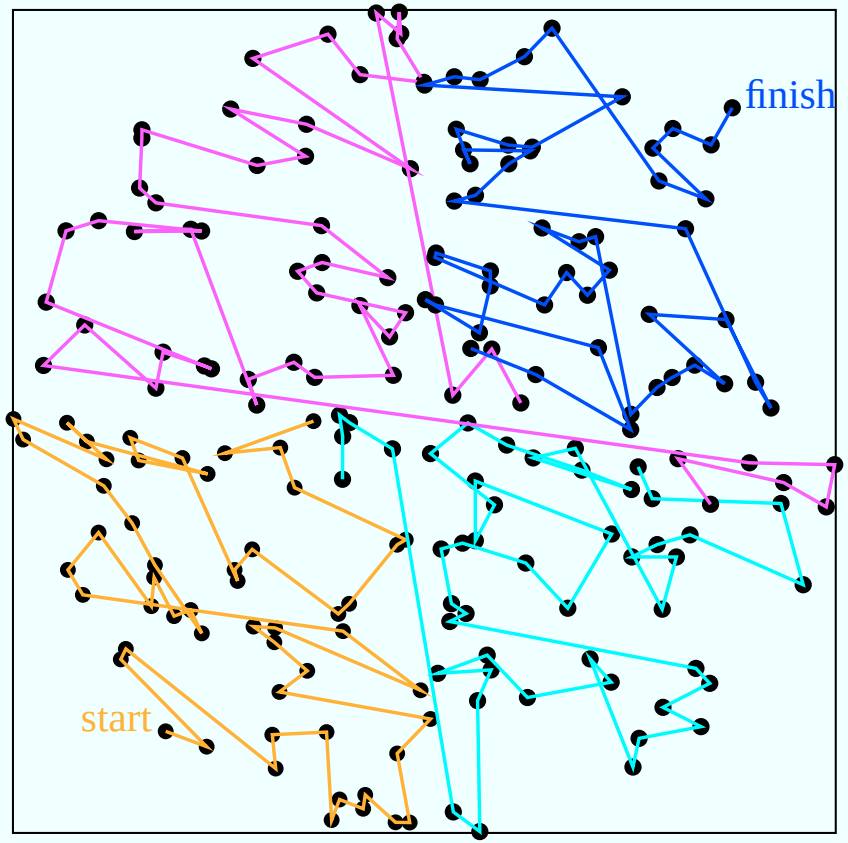
Smallest box length $s = L/2^{\text{nlevel}} = 0.5$

Integer coords:

Real r_i (z, y, x):	4.0	3.5	2.5
Integer r_i/s:	8	7	5
Binary:	01000	00111	00101

Interleave bits: 1 000 100 011 010 011

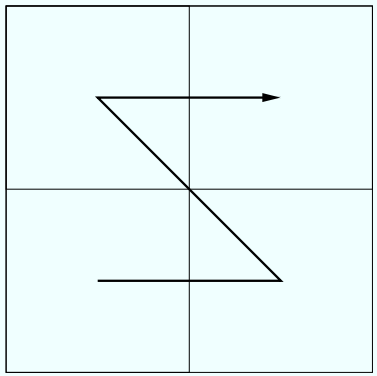
On 64-bit machine have 21 bits/coordinate + placeholder
$\rightarrow$ max. 20 refinement levels with signed integers.

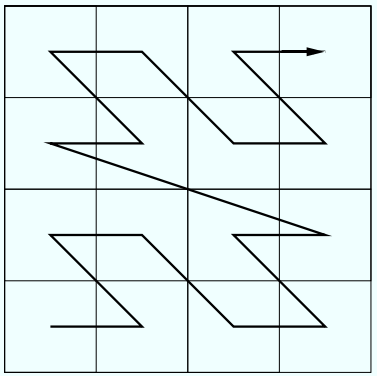


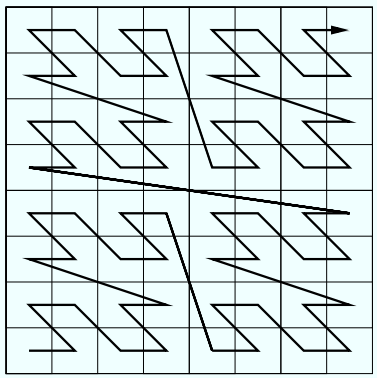
Global key-sort maps 3D particle coordinates onto 1D space-filling curve. Dividing curve into equal segments provides natural domain decomposition, with possibility to adjust weighting according to operation count in force computation.

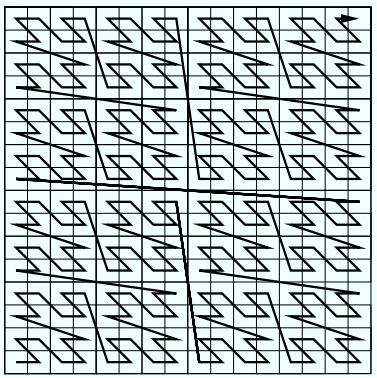
A continuous function $f : \mathcal{I} \to \mathbb{R}^n$ of the compact set $\mathcal{I} \subset \mathbb{R}$ into $\mathbb{R}^n$ ($n \geq 2$) is called space filling curve if its image $f_*(\mathcal{I})$ has a Jordan content (area, volume, ...) > 0.

Morton (Z)

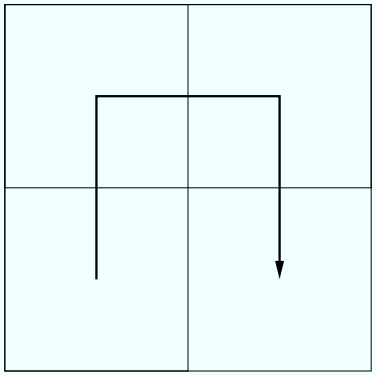


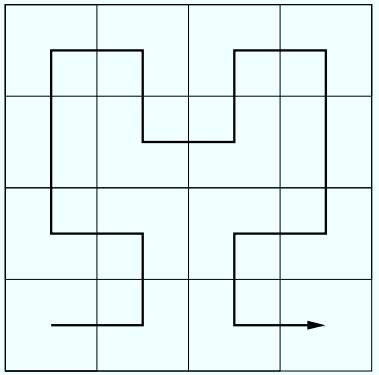


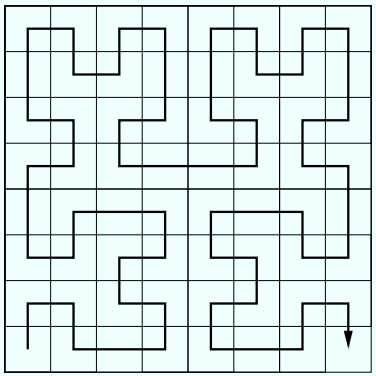


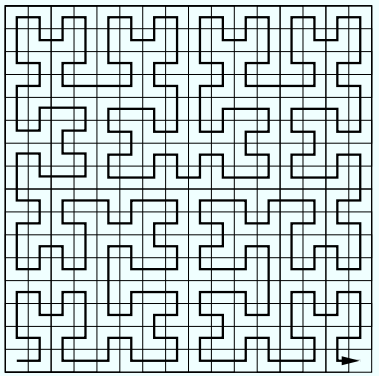


Peano-Hilbert





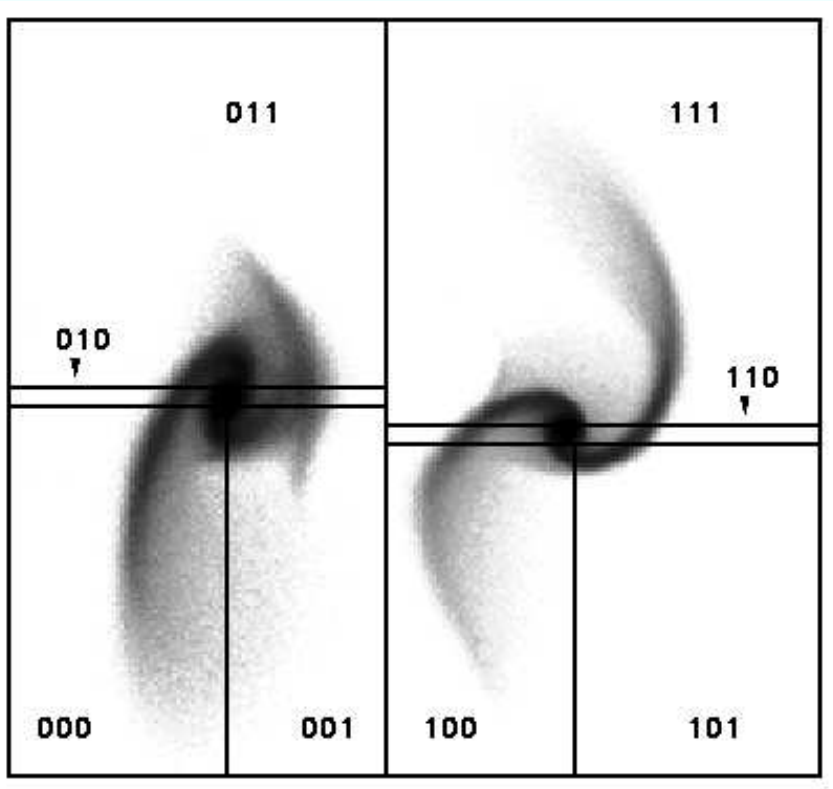




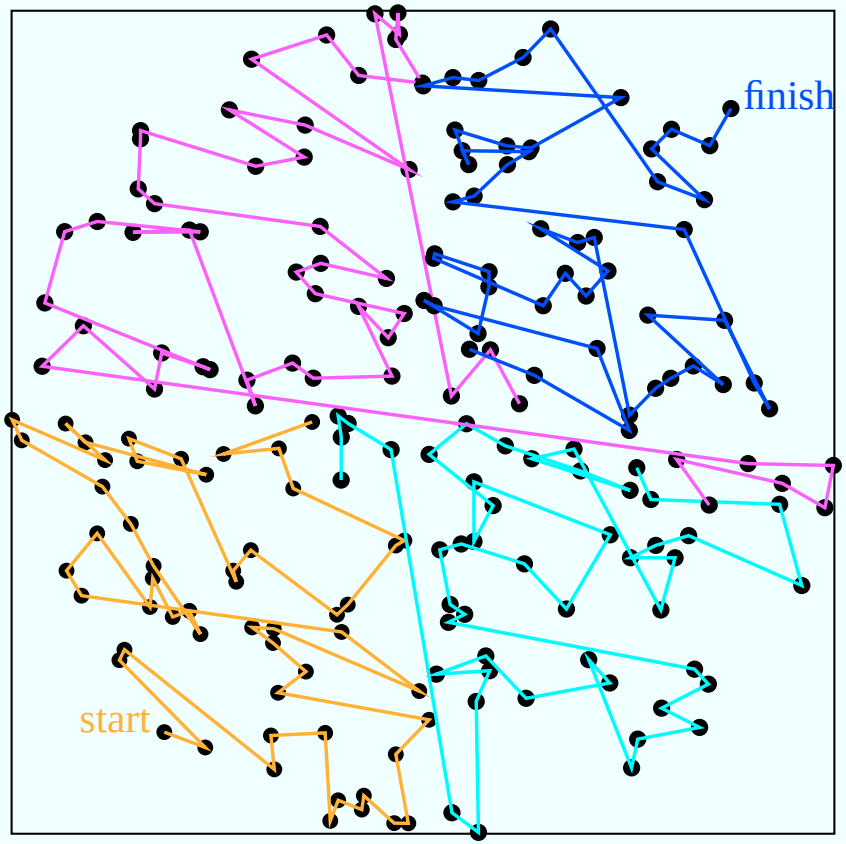
25 – ... and in 3D:

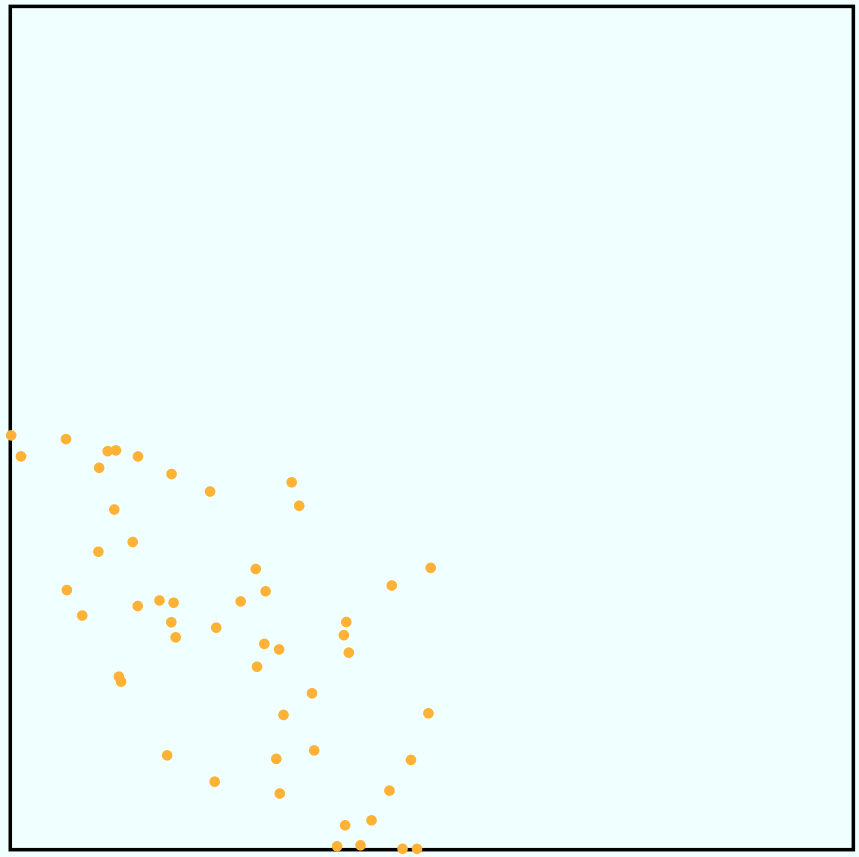
25 – Alternative method: Orthogonal Recursive Bisection 195/206

- Uses binary tree
- Freedom to choose dimension and position of subdivision, provided that equal amounts of work remain for subvolumes
- Finally get 2^n subvolumes for n-level tree



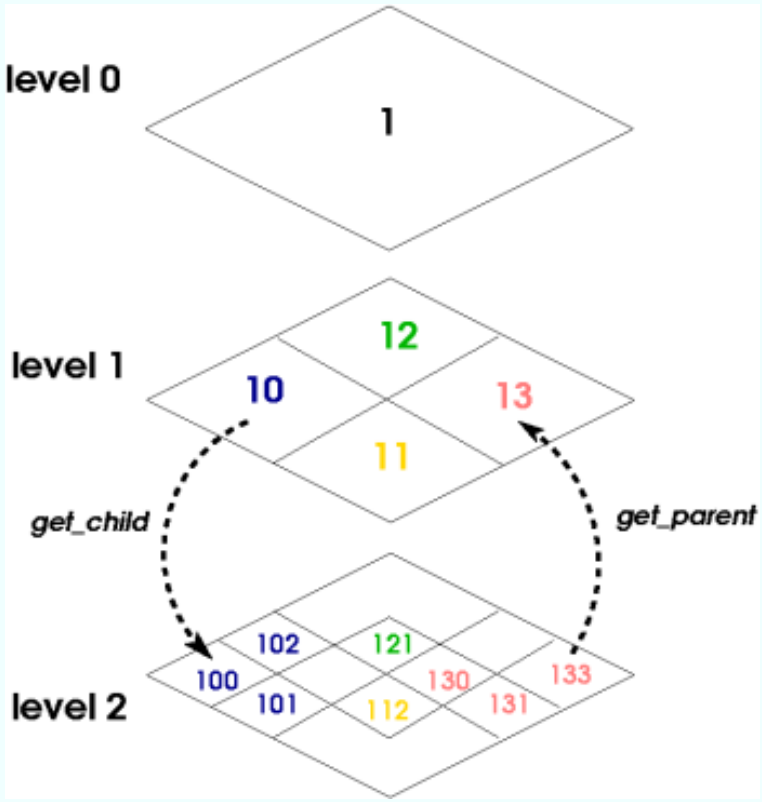
- See: Dubinsky, New Astronomy 1, 133 (1996)





Local particles on Processor 0

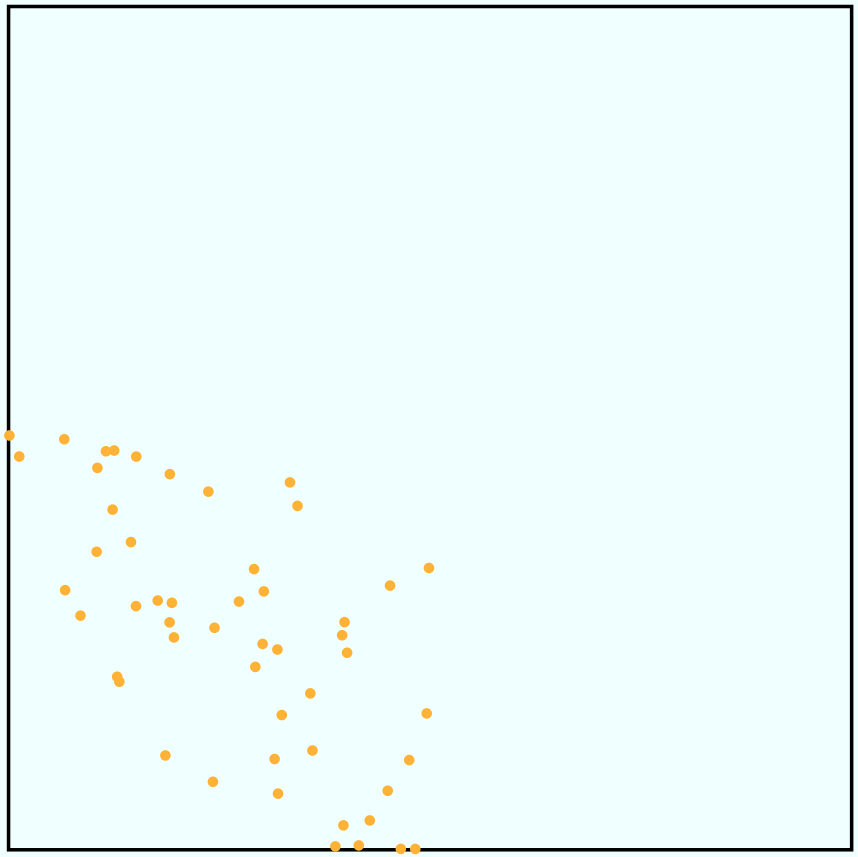
25 – Key operations (modulo 4)

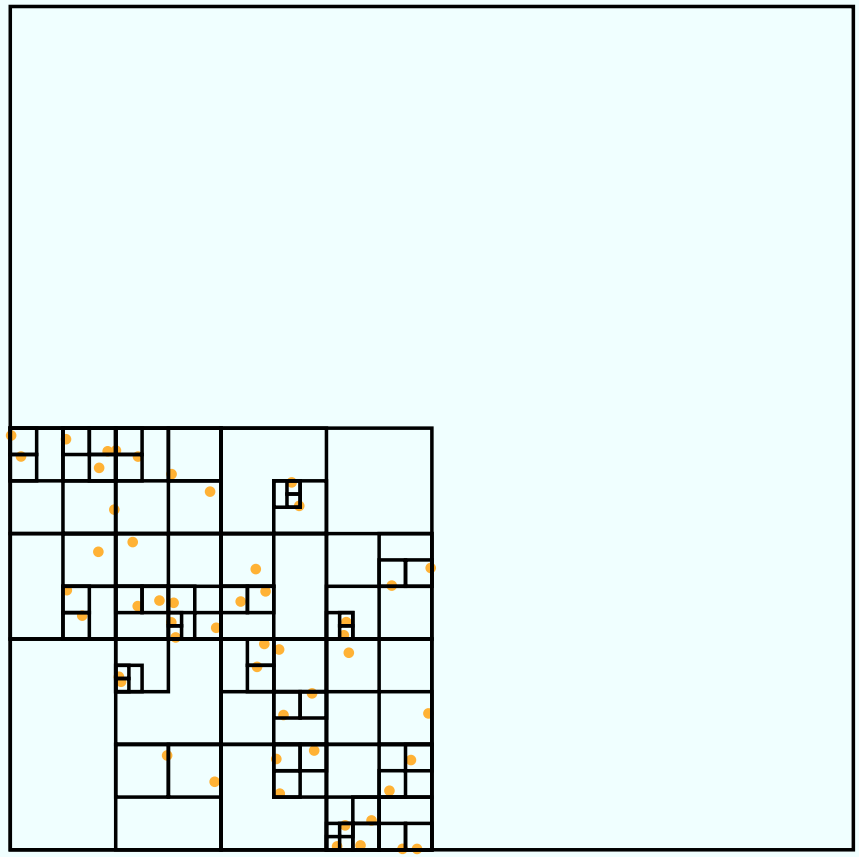


Parent node: strip right 3 bits

Child node: left shift + child-code (0-7)

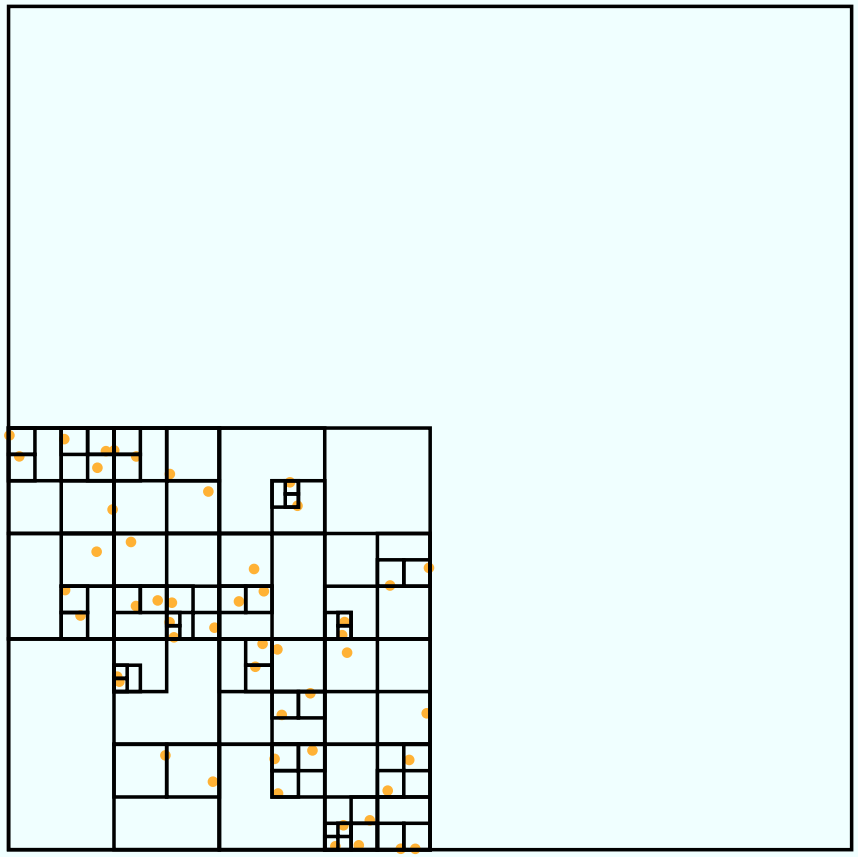
25 – Local tree build

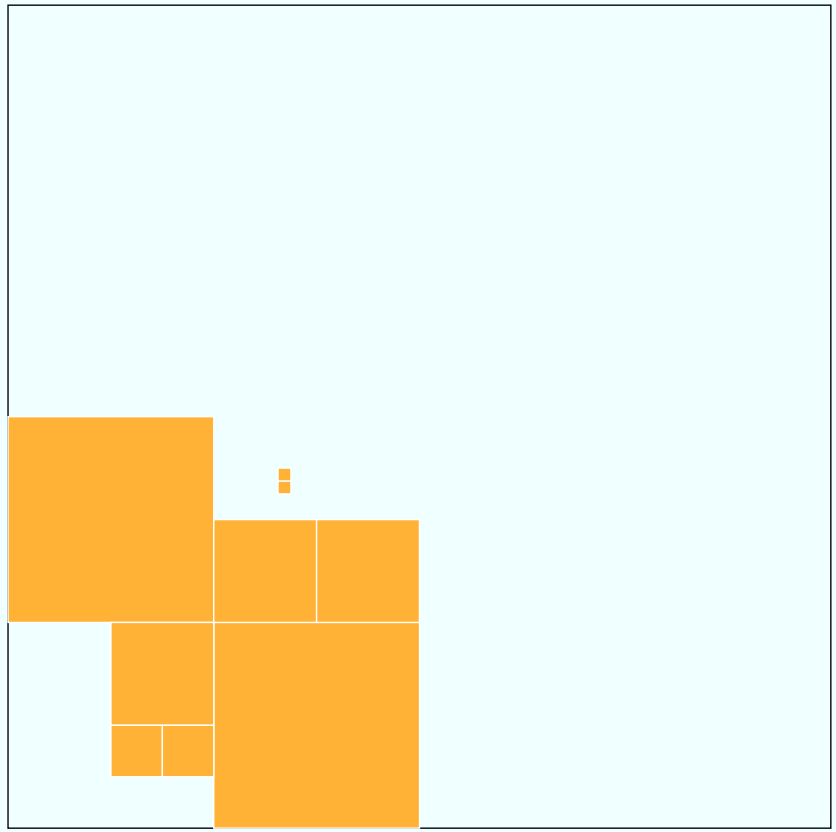




Local tree nodes can be derived immediately from particle keys using bit-shift operations.

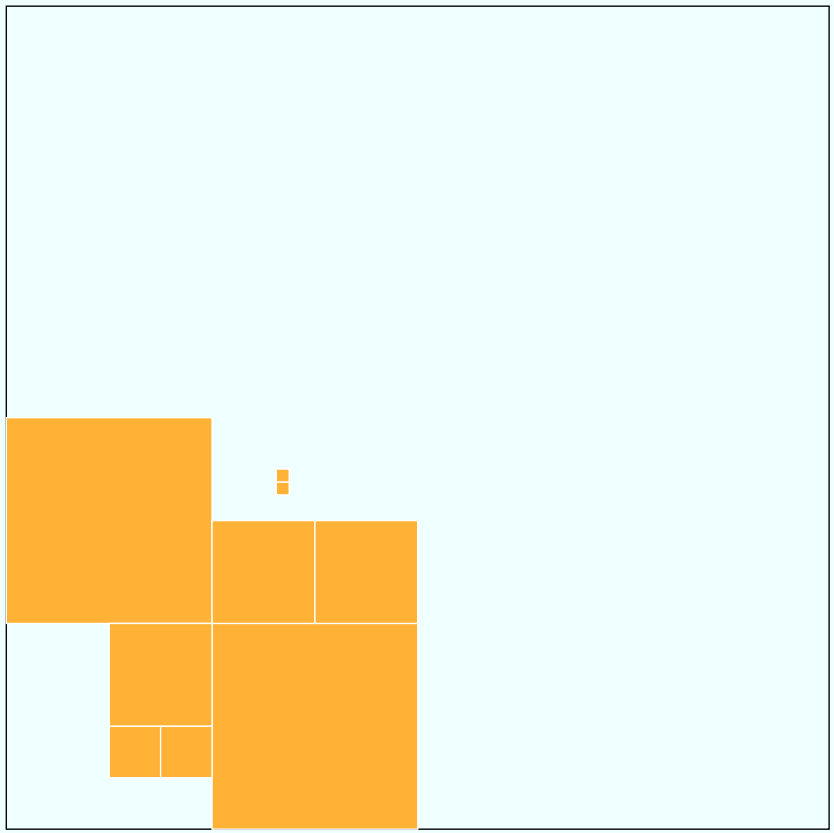
25 – Define local ’branch’ nodes

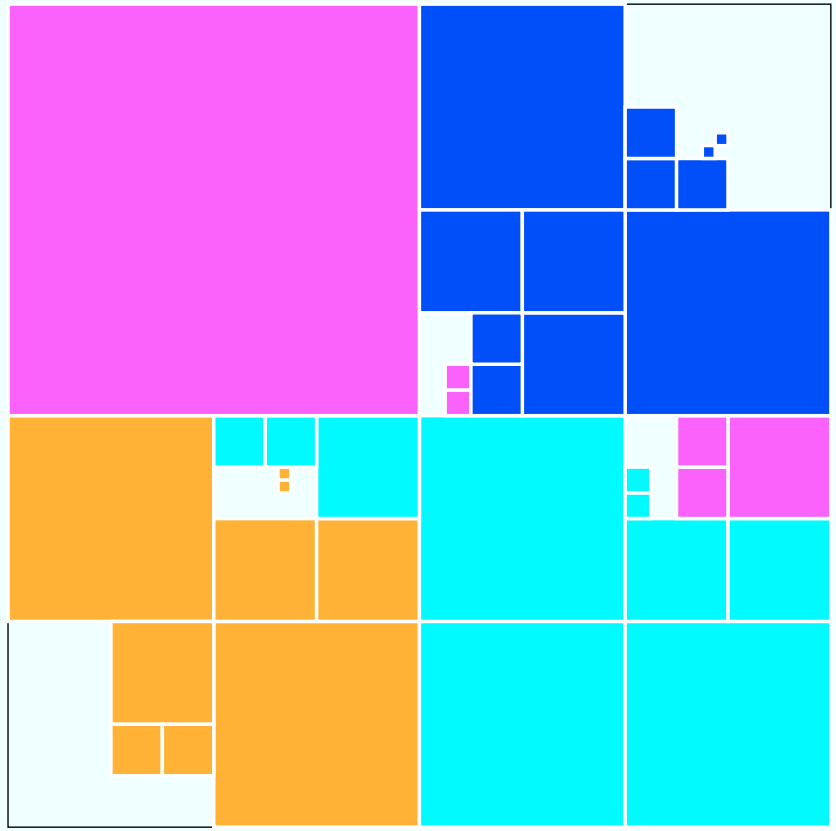




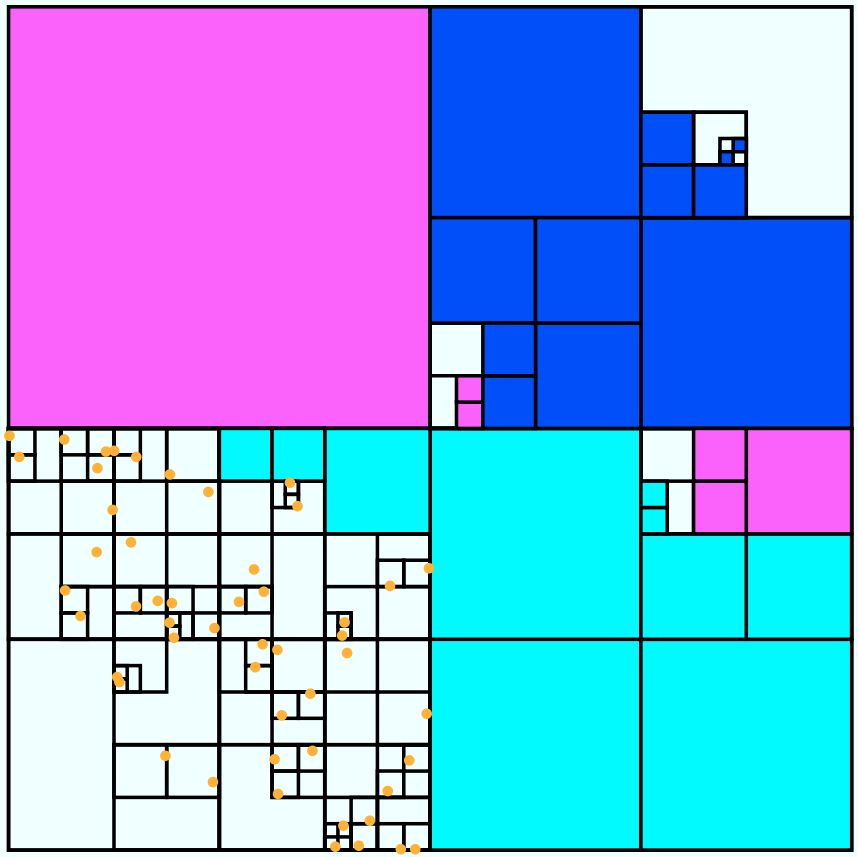
Smallest set of nodes which includes all local particles

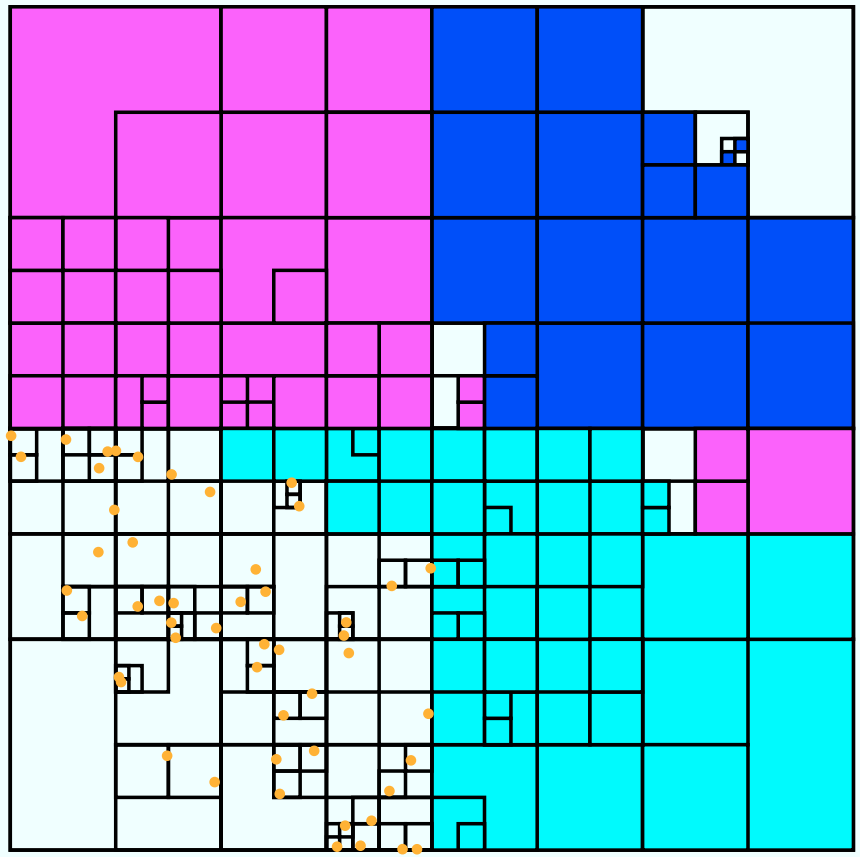
25 – Exchange to create global branch nodes



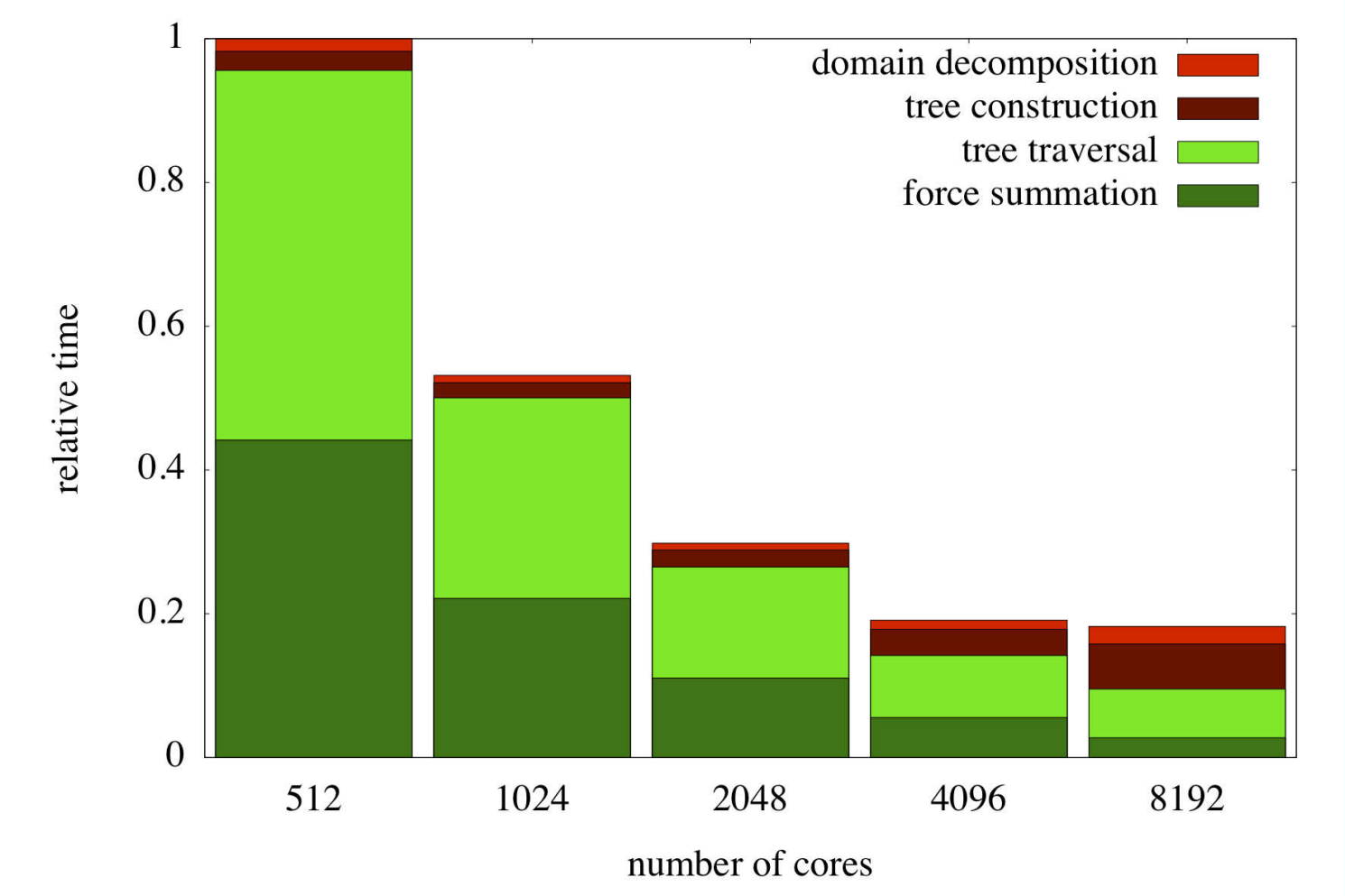


All-to-all swap of branch nodes creates entry points for the tree traversal on each processor.

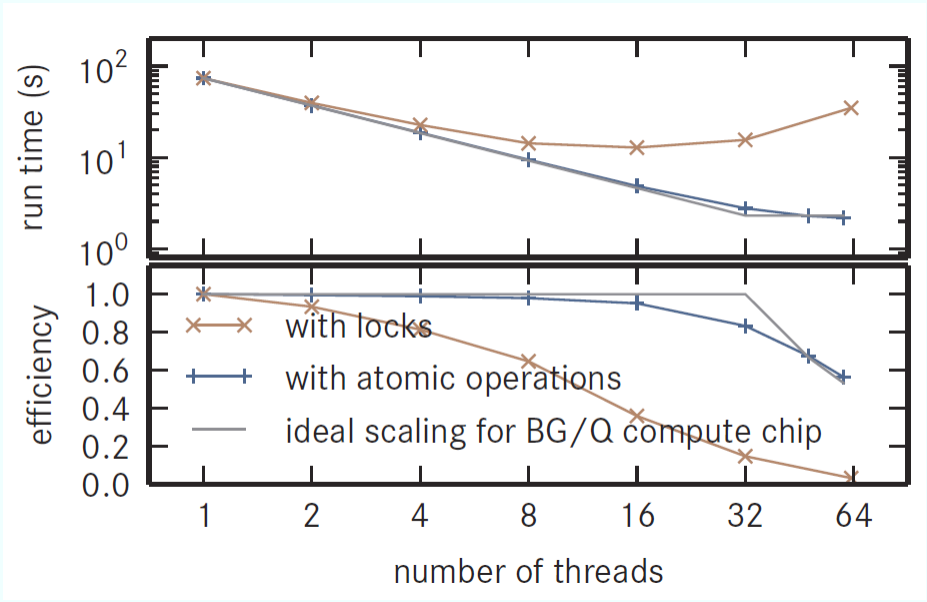


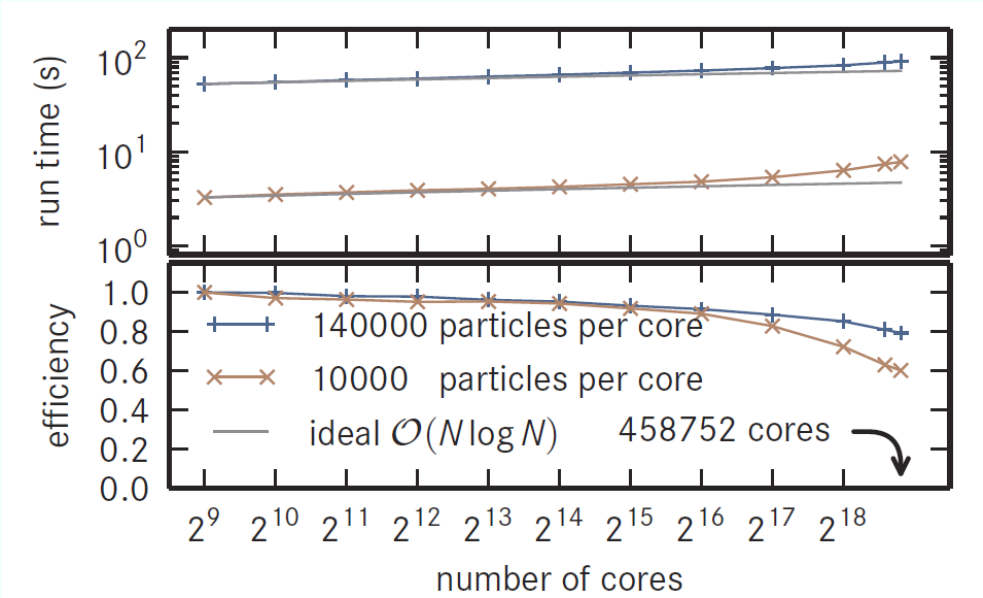


Tree information on processor 0 before and after tree traversal. Nodes required (by MAC) for force summation are copied into local tree.


domain decomposition
tree construction
tree traversal
force summation
relative time
1
0.8
0.6
0.4
0.2
0
512
1024
2048
4096
8192
number of cores

25 – Parallel scaling: hybrid MPI-pthread version





26 – Outline

26 – Practice with space-filling curves

- For the disc or box particle distributions used in one of the first assignments (A1/2), create a 2D or 3D space-filling curve according to the Morton (Z) ordering illustrated earlier.
- Hint 1: Use bit operators to extract and piece together the particle keys. In Fortran the (2D) operation uses the `ibits` operator, like this:
  key = place + SUM((/ (4**i*(2*ibits(iy,i,1) + ibits(ix,i,1)),i=0,nbits-1) /))
- Hint 2: to generate the curve, you will need to sort your particle keys with a standard sorting algorithm (bubble, heap, radix etc) - see Numerical Recipes or Python script simple_sort.py provided.
- Now try a non-uniform particle distribution - display your results graphically.
- (optional) Modify the particle keys to give the Peano-Hilbert curve.
  Hints here (C code):
  `http://en.wikipedia.org/wiki/Hilbert_curve`
  and here (Python):
  `https://github.com/dentearl/simpleHilbertCurve/blob/master/src/simpleHilbertCurve.py`
- (optional) Which curve do you think is better for domain decomposition and why?

- M. McCool, A. D. Robison, J. Reinders, *Structured Parallel Programming*, Elsevier (2012)
- G. Sutmann et al., *Comparison of scalable fast methods for long-range interactions*, Phys. Rev. E **88**, 063308 (2013).
- M. Winkel et al., *A massively parallel, multi-disciplinary Barnes–Hut tree code for extreme-scale N-body simulations*, Comp. Phys. Commun. **183**, 880 (2012).