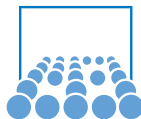


Scientific Computing II

Molecular Dynamics Simulation (4)

Michael Bader – SCCS

Summer Term 2013



Numerical Methods for Long-Range Potentials

Introduction

Tree-Based Methods

The Barnes-Hut Method

Numerical Methods for Long-Range Potentials

Short-range potentials (Lennard-Jones, e.g.):

- resulting mutual interactions are restricted to particles in some local neighbourhood
- facilitates numerical treatment and algorithmic organization: no quadratic complexity induced by an "all-to-all" behaviour

Long-range potentials:

- examples: Coulomb or gravity potential
- interactions between remote particles must not be neglected
- simple cut-off not possible
- nevertheless need for approaches that avoid quadratic complexity

Long-Range Potentials (2)

What is long-range?

- intuitively: potential function $U(r)$ does not decrease rapidly with increasing r
- formally (one possibility): for $d > 2$, potentials not decreasing faster than r^{-d} for increasing r (criterion: integrability over \mathbb{R}^d)

In practice:

- typical potentials in applications have both – a short-range part (to be dealt with according to the previous sections) and a long-range part,
- represented as two additive components:

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

Mesh-Based Methods

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

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

Examples for mesh-based methods:

- P³M (Particle-Particle–Particle-Mesh) method
- PME (Particle-Mesh-Ewald) method
- SPME (Smooth-Particle-Mesh-Ewald) method)

Mesh-Based Methods

PDE Formulation for potential Φ :

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

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

Hierarchical or Tree-Based Methods

- starting point: integral representation of the potential Φ

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

- advantageous especially for heterogeneous particle distributions (frequent in astrophysics, relevant also for molecular dynamics)
- examples:
 - panel clustering
 - Barnes-Hut method
 - (fast) multipole methods

Tree-Based Methods

- based on integral representation of the potential
- hierarchical decompositions of the domain of simulation
- adaptive approximation of the particle distribution
- widespread scheme: **octrees**
- allow for separation of **near-field** and **far-field** influences
- log-linear or even linear complexity can be obtained
- high flexibility with respect to more general potentials (as needed for special applications, such as biomolecular problems)

Series Expansion of the Potential

- general (integral) representation of the potential:

$$\Phi(x) = \int_{\Omega} G(x, y) \varrho(y) dy$$

(general kernel G , particle density $\varrho(y)$, and domain Ω)

- Taylor expansion of the kernel G in y around y_0 :

$$G(x, y) = \sum_{\|j\|_1 \leq p} \frac{1}{j!} G_{0,j}(x, y_0) (y - y_0)^j + R_p(x, y)$$

- multi-index $j = (j_1, j_2, j_3)$, $j! = j_1! j_2! j_3!$
- $G_{k,j}(x, y)$ mixed (k, j) -th derivative
- remainder $R_p(x, y) \rightsquigarrow$ skipped higher order terms

Series Expansion of the Potential (2)

- leads to expansion (and approximation) of the potential:

$$\Phi(x) \approx \sum_{\|j\|_1 \leq p} \frac{1}{j!} M_j(\Omega, y_0) G_{0,j}(x, y_0)$$

with the so-called **moments**

$$M_j(\Omega, y_0) := \int_{\Omega} \varrho(y) (y - y_0)^j dy$$

Subdivision of the Domain

- separation of near-field and far-field for given x :

$$\Omega = \Omega^{\text{near}} \cup \Omega^{\text{far}}, \quad \Omega^{\text{near}} \cap \Omega^{\text{far}} = \emptyset$$

- decompose far-field into disjoint, convex subdomains:

$$\Omega^{\text{far}} = \bigcup_i \Omega_i^{\text{far}}$$

- note: decomposition depends on x , i.e. it is done for each particle position x
- each Ω_i^{far} has an associated point y_0^i
- how to choose the subdivision?

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

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

Subdivision of the Domain (2)

resulting approximation for $\Phi(x)$:

$$\begin{aligned}
 \Phi(x) &= \int_{\Omega} \varrho(y) G(x, y) dy \\
 &= \int_{\Omega^{\text{near}}} \varrho(y) G(x, y) dy + \int_{\Omega^{\text{far}}} \varrho(y) G(x, y) dy \\
 &= \int_{\Omega^{\text{near}}} \varrho(y) G(x, y) dy + \sum_i \int_{\Omega_i^{\text{far}}} \varrho(y) G(x, y) dy \\
 &\approx \int_{\Omega^{\text{near}}} \varrho(y) G(x, y) dy + \sum_i \sum_{\|j\|_1 \leq p} \frac{1}{j!} M_j(\Omega_i^{\text{far}}, y_0^i) G_{0,j}(x, y_0^i)
 \end{aligned}$$

Error Estimates

- error characteristics for one fix particle position $x \in \Omega$:
 - local relative approximation error for one Ω_i^{far} can be shown to be of order $O(\theta^{p+1})$
 - global relative approximation error (summation over whole far-field) can be shown to be of order $O(\theta^{p+1})$
- clarifies the role of θ :
 - control the global approximation error in x
 - geometric requirement for the far-field subdivision: the closer Ω_i^{far} is to x , the smaller it has to be to satisfy the θ -condition
- hence: a typical "level of detail" approach
 - the closer, the higher resolved
 - cf. terrain representation in flight simulators

Tree Structures

Central question:

How can we construct all these necessary separations of near-fields and far-fields and subdivisions of far-fields in an efficient way?

Idea:

- recursive decomposition of Ω (square in 2D, cube in 3D, ...) into cells of different size,
- terminate the subdivision process if a cell is either empty or contains just one particle

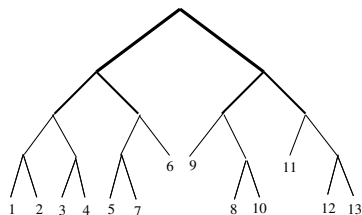
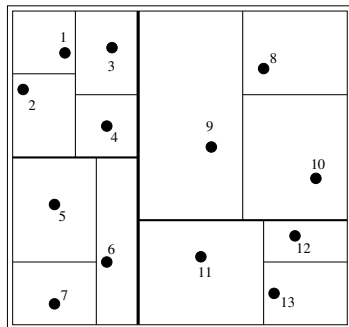
Tree Structures

Concepts:

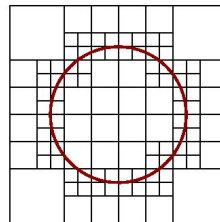
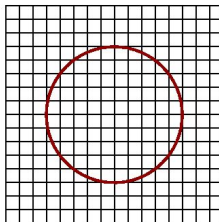
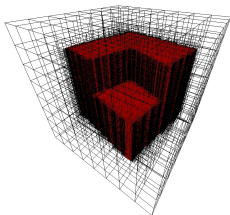
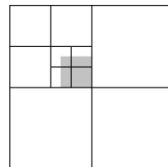
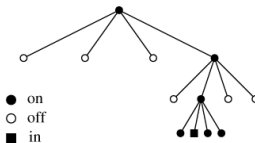
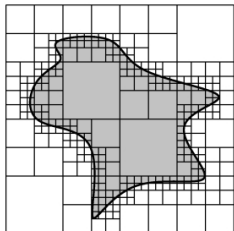
- **kd-tree**: alternate subdivision in coordinate direction (x , y , and z), such that the separation produces two subdomains that roughly contain the same number of particles each
- **quadtree** (2D) or **octree** (3D): subdivision into four congruent subsquares or eight congruent subcubes, respectively

The following algorithms (Barnes-Hut etc.) use the octree approach

kd-Trees – Example



Quadrees and Octrees – Examples



Recursive Computation of the Far-Field Subdivision

- create the octree corresponding to the set of particles
 - each node of the octree represents a subdomain of Ω or one **cell**
 - for each cell i , define some y_0^i (the centre point or the centre of gravity of all particles contained, e.g.) for doing the Taylor expansion
 - for each cell i , let the parameter diam just denote the diameter of the smallest surrounding sphere, e.g.
- objective: for each particle position x , use as few cells as possible (i.e. as big cells as possible) for fulfilling the $\text{diam}-\theta$ rule

Recursive Computation of the Far-Field Subdivision (2)

- hence: start from root node, check

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

stop if fulfilled (no need for further subdivision) and proceed if not yet fulfilled

- note that for each x , we typically get a different subdivision
- but note also that all these subdivisions are just subtrees of our constructed octree

Recursive Computation of the Moments

- now: use this subdivision for the efficient calculation of the local moments $M_j(\Omega_i^{\text{far}}, y_0^i)$
- direct (numerical) integration or direct summation are not efficient
- therefore: use hierarchical tree structure to calculate all moments for all cells in one run and store them
- crucial property for that:

$$M_j(\Omega_1 \cup \Omega_2, y_0) = M_j(\Omega_1, y_0) + M_j(\Omega_2, y_0),$$

if the point of expansion y_0 is the same

Recursive Computation of the Moments (2)

- in the (standard) case of different y_0^1 and y_0^2 , there are simple conversion formulas:

$$M_j(\Omega_1, \hat{y}_0) = \sum_{k \leq j} \binom{j}{k} (y_0 - \hat{y}_0)^{j-k} M_k(\Omega_1, y_0)$$

($k \leq j$ component-wise, multiplicative binomial coefficients)

- this allows for a bottom-up calculation of the moments from the leaves to the root
- In the leaves:
 - if no particle present: zero
 - if one particle of mass m there in x : $m(x - y_0^i)^j$

Using these Building Blocks

Still to be done for a numerical routine:

- how to construct the tree, starting from a given set of particles?
- how to store the tree?
- how to choose cells and expansion points?
- how to determine far-field and near-field?

Several algorithmic variants exist

→ original Barnes Hut algorithm described in the following

The Barnes-Hut Method

- oldest (dates back to 1986), simplest, and most widespread hierarchical tree-based approach
- original target applications: astrophysics (high particle numbers, very heterogeneous density distribution)
- particle-particle interaction via gravity potential:

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

- uses octrees: leaves of the tree represent empty cells or one particle, inner nodes represent clusters of particles or **pseudo particles**
- main idea: approximate gravity induced by particles in a remote cell by the force induced by one pseudo particle (with resp. accumulated mass in the cell's centre of gravity)

The Barnes-Hut Method (2)

Three main steps:

1. construction of the octree (top-down recursion):
refine, until just one or no particle exists per cell
2. calculation of pseudo particles (bottom-up recursion):
mass is just sum of the cell's particles' masses,
associated point is the mass-weighted average of the
particles' positions
3. calculation of forces (N incomplete top-down traversals for
 N particles)

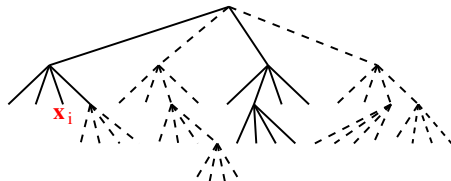
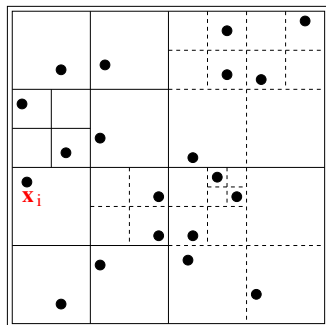
Calculation of the Forces

- remember: to be done for each particle
- assume a particle with position $x \in \Omega$
 - start in root node
 - proceed to son nodes, until the θ – rule: $\frac{\text{diam}}{r} \leq \theta$ is fulfilled, where r denotes the distance of the corresponding pseudo particle to x
 - then, add the resulting interaction influence to the overall result
- implicit separation into near-field and far-field:
 - near-field: all leaves reached during this traversal (single-particle influence)
 - far-field: all inner nodes where the process stops, i.e. cells representing pseudo particles

Calculation of the Forces (2)

- several variants concerning the determination of the parameter $diam$
- Barnes-Hut method can be interpreted as a special case of the Taylor approximation discussed above for $p = 0$ (for the calculation of the pseudo particles, we just sum up masses, i.e. zero-th moments)

Example of the Tree Construction and Force Calculation



Accuracy and Complexity

Accuracy of Barnes Hut:

- depends on control parameter θ
- the smaller we choose θ , the larger gets the near-field (and thus: smaller error from far-field approximation)
- however, slow $\mathcal{O}(\theta)$ convergence due to $p = 0$

Complexity:

- increases for decreasing θ
- for roughly homogeneous particle distributions:
 - number of active cells bounded by $C \log N / \theta^3$ for some constant C and N particles
 - overall cost of order $\mathcal{O}(\theta^{-3} N \log N)$
- for limit case $\theta \rightarrow 0$: method degenerates to $\mathcal{O}(N^2)$ complexity of the original “all-to-all” approach

Some Remarks on the Implementation

- use a standard data structure for each (pseudo) particle
- standard tree implementation with pointers:
 - each node contains one (pseudo) particle at most
 - subdomain corresponding to a node/cell can be either stored explicitly or calculated on-the-fly during a tree traversal
 - four (2D) or eight (3D) pointers from a node to its sons
- linearisation is possible, too (i.e. no need for pointers)
- tree traversal typically by recursion
 - pre-order (top-down)
 - post-order (bottom-up)

Some Remarks on the Implementation (2)

- tree construction:
 - successive insertion of particles
 - starting from the empty tree (i.e. the octree consisting of the root node only)
 - refine, when necessary (i.e. two particles in one node)
- calculation of pseudo particles:
 - bottom-up (post-order) traversal
 - sum of masses, weighted average of positions as described before

Some Remarks on the Implementation (3)

- calculation of forces:
 - outer loop: traversal visiting each leaf (to get the far-field approximation for each particle)
 - inner loop: top-down (pre-order) traversal until the θ -rule is fulfilled
 - parameter diam on-the-fly
- time integration: essentially as before, now as another tree traversal
- movement of particles within the tree:
 - either via constructing a new octree
 - or via modifying the existing octree (generally preferred)