Long-range Interactions in Particle Simulations
ScaFaCoS

Olaf Lenz
Outline

- Many-particle Simulations
- Short-range Interactions
- Long-range Interactions: ScaFaCoS
- If time permits: ESPResSo and coarse-graining

- Many slides from Axel Arnold (ICP Stuttgart) and Godehard Sutmann (FZ Jülich)
- Algorithms, but not so much mathematics (too short time, and I'm too bad)
Many-particle Simulations

- Many-particle simulations are a standard tool in many branches of science
  - Plasma Physics
  - Physical Chemistry
  - Biochemistry and -physics ("Molecular Dynamics" MD)
  - Soft Matter
  - Material sciences
  - Astrophysics and Cosmology
The New Trinity of Physics?

Experiment

Computer Simulations (a.k.a. “Computer Experiments”)

Theory
Usage of FZ Jülich Supercomputers

**Tier-0: Jügene**
Blue Gene/P

**Tier-1: Jüropa**
Bull/Sun Nehalem-Cluster

- **Astrophysics**
- **Biophysics**
- **Chemistry**
- **Earth & Environment**
- **Plasma Physics**
- **Soft Matter**
- **Fluid Dynamics**
- **Elementary Particle Physics**
- **Computer Science**
- **Condensed Matter**
- **Material Science**

Olaf Lenz

ScaFaCoS and ESPResSo
How To Do a Many-Particle Simulation?

- **System:**
  - *N Particles at positions* \( \vec{x}_i \)
  - \( \sim O(N^2) \) (Pair) Interactions between them

- **Numerically integrate Newton's Equations of Motion:**
  \[
  \ddot{\vec{x}}_i = \frac{f(\vec{x}_i)}{m}
  \]

- Involves computing \( f(\vec{x}_i) \) and propagation of \( \vec{x}_i \) in every timestep

- **Computational cost of naïve implementation:**
  - *Propagating the particles:* \( O(N) \)
  - *Computing the Interactions* \( f(\vec{x}_i) \): \( O(N^2) \)
Example: All-Atom Molecular Dynamics

- e.g. AMBER, GROMACS, NAMD, …
- Interactions (AMBER/Sander Force Field):

\[
\mathcal{U} = \sum_{\text{bonds}} K_r (r - r_{eq})^2 + \sum_{\text{angles}} K_\theta (\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2} [1 + \cos(n\phi - \gamma)] \\
+ \sum_{i<j} \left[ \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6} \right] + \sum_{\text{H-bonds}} \left[ \frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right] + \sum_{i<j} \frac{q_i q_j}{r_{ij}}
\]

- "Lennard-Jones"
- H-bonds
- Coulomb interaction
- van-der-Waals interaction
- Pauli exclusion
Short-range vs. Long-range Interactions

- Many interactions are *short-ranged*
- i.e. it is possible to cut off the interaction at a certain distance
- Formal definition:

\[
I = \int_{\mathbb{R}^D} d^D r \frac{A}{r^n} = \begin{cases} \infty : n \leq D & : \text{long range} \\ \text{finite} : n > D & : \text{short range} \end{cases}
\]

- Revisiting AMBER:

\[
U = \sum_{\text{bonds}} K_r (r - r_{eq})^2 + \sum_{\text{angles}} K_\theta (\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[ \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6} \right] + \sum_{\text{H-bonds}} \left[ \frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right] + \sum_{i<j} \frac{q_i q_j}{r_{ij}}
\]

short-range \hspace{5cm} \text{long-range}
Speeding Up Short-range Interactions: Cell lists

- a.k.a. Linked Cell Algorithm
- Split the simulation box into Cells with side length $r_{cut}$
- Create and maintain a list of particles in each cell
- Only need to compute the interactions with particles in a neighboring cell
- Good news: Reduces time complexity to $O(n)!$ (at constant density)

Speeding Up Short-range Interactions: Verlet lists

- For each particle, store a list of particles in a distance $< r_s$
- Verlet lists need to be updated when a particle travelled further than $\frac{1}{2}$ skin
- Reduces number of interaction partners by $\sim 40\%$ compared to Cell lists
- Use on top of Cell lists
- Significant memory requirements
- Most probably not a very good algorithm in the future!

Parallelizing Short-range Interactions

- Nowadays rarely “Atomic Decomposition”
- Typically Domain Decomposition
- Next-neighbor communication
- Size of border region $r_{\text{cut}}$
- Can employ Cell lists data structures
- Various tricks of the trade, Improved algorithms, Load balancing
- Still not very good for exascale computing
  - *Domains should not become smaller than* $r_{\text{cut}}$
  - *Requires a few hundred particles per process to be efficient*

Stolen from W. Smith, Daresbury Lab, Warrington, UK
Periodic Boundary Conditions

Simulated systems are much smaller than “real” systems
  • *Boundaries make up a significant part of the system!*

- Trick: *Periodic Boundary Conditions*
- The simulated system has infinitely many copies of itself in all directions
- A particle at the right boundary interacts with the particle at the left boundary in the image
- *Minimum image convention*: Each particle only interacts with the closest image of another particle
- Pseudo-infinite system without boundaries
- Significantly reduces boundary-related errors
Long-range Interactions

For long-range interactions (LRI), these tricks do not work!

- Why can't we just cut off anyway?
  - Qualitatively different behavior due to long-range interactions, e.g.
    - Charge inversion
    - Like-charge attraction

- Direct summation: O(N²)

- Worse in PBC
  - Need to take into account several periodic images
  - Slowly convergent sum

Coulomb Potential

\[ U = \frac{1}{2} \sum_{S=0}^{\infty} \sum_{m \in S} \sum_{i,j=1}^{N} \frac{q_i q_j}{|r_{ij} + m L|} \]

or

\[ U = \frac{1}{2} \sum_{i=1}^{N} q_i \phi(r_i) \]

\[ \nabla^2 \phi(r) = 4\pi \sum_{j=1}^{N} \delta(r_j - r) q_j \]

(Poisson Equation)

Like-charge attraction
Methods for Long-range Interactions

Luckily, a number of smart algorithms do exist

- Unfortunately, they are complex
- Often they do not parallelize easily
- Prefactors and parallel scaling are highly platform and implementation dependent
- LRI often account for a significant part of the computational cost

<table>
<thead>
<tr>
<th>Year</th>
<th>Method</th>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1820</td>
<td>Direct Summation</td>
<td>$O(N^2)$</td>
<td>Laplace</td>
</tr>
<tr>
<td>1921</td>
<td>Ewald Summation</td>
<td>$O(N^{3/2})$</td>
<td>Ewald</td>
</tr>
<tr>
<td>1977</td>
<td>Multigrid Summation</td>
<td>$O(N)$</td>
<td>Brandt</td>
</tr>
<tr>
<td>1986</td>
<td>Barnes-Hut Treecode</td>
<td>$O(N \log N)$</td>
<td>Barnes, Hut</td>
</tr>
<tr>
<td>1987</td>
<td>Fast Multipole Method</td>
<td>$O(N)$</td>
<td>Greengard, Rokhlin</td>
</tr>
<tr>
<td>1988</td>
<td>Particle Mesh</td>
<td>$O(N \log N)$</td>
<td>Hockney, Eastwood</td>
</tr>
<tr>
<td>1993</td>
<td>Particle Mesh Ewald</td>
<td>$O(N \log N)$</td>
<td>Darden</td>
</tr>
</tbody>
</table>
Enter: ScaFaCoS

“Scalable Fast Coulomb Solvers”
- Library that provides various of these algorithms
- Common interface
- Focus on scalability
- Financial support by BMBF HPC-1 Call
- Coordinated by the Forschungszentrum Jülich (IAS/JSC)
- University Partners: Bonn, Chemnitz, Stuttgart, Wuppertal
- Research Centres: MPIP Mainz, Fraunhofer St. Augustin
- Industry: BASF, Cognis, IBM
- Alas, not finished yet!
  - Only preliminary results
  - Expect publication in Q2 2012
Ewald Summation

- Not implemented in ScaFaCoS, just for understanding
- Useful mostly for Periodic Boundary Conditions

Two problems with Coulomb Interaction
- Slowly decaying, long tail
- Singular at each particle position

Idea: Separate the problems: Charge distribution

- "Far field"
  - Smear out charges with a Gaussian
  - Smooth distribution in periodic BC
  - Solve Poisson Equation in reciprocal space

- "Near field"
  - Subtract the effect of the far field
  - Yields in short-ranged interaction with cut-off
  - Compute directly in real space

P. P. Ewald, 1888 - 1985
Ewald Summation

- **Near field**
  - Compute short-ranged near field interaction

- **Far field**
  - Fourier transform smeared out charge distribution
  - Cut off in Fourier space
  - Solve Poisson Equation in reciprocal space via Green’s Function
  - Backtransform potential to real space
  - Differentiate potential to get fields or forces

- Overall complexity: $O(N^{3/2})$

---

Ewald sum

$$U = U^{(r)} + U^{(k)} + U^{(s)}$$

with

$$U^{(r)} = \frac{1}{2} \sum_{m \in \mathbb{Z}^3} \sum_{i,j} q_i q_j \text{erfc}(\alpha |r_{ij} + mL|)$$

$$U^{(k)} = \frac{1}{2L^3} \sum_{\mathbf{k} \neq 0} \frac{4\pi}{k^2} e^{-k^2/4\alpha^2} |\hat{\rho}(\mathbf{k})|^2$$

$$U^{(s)} = -\frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2$$

forces from differentiation

$$F_i = -\frac{\partial}{\partial r_i} U$$
P³M/PME: Ewald Summation on a Mesh

- Idea: Use discrete FFT (O(N log N)) instead of normal Fourier Transform
- Requires a regular mesh
- Far field
  - Assign charges onto the mesh
  - Forward-FFT mesh
  - Solve Poisson Equation by applying Green's function ("Influence function")
  - Back-FFT mesh
  - Interpolate potentials onto charge positions
  - Force computation
    - Differentiate potential in Fourier space (ik Differentiation), then backtransform and interpolate forces (3xFFT, high precision)
    - Or backtransform potential, then differentiate and interpolate analytically in real space (1xFFT, lower precision)
- Near field as Ewald Summation
P³M/PME: Ewald Summation on a Mesh

- Particle-Particle-Particle-Mesh “P³M” (Hockney/Eastwood 1988)
- Particle-Mesh-Ewald “PME” (Darden 1993)
- Similar algorithms, differences in the details
- Used for a long time, standard method in MD
- Good error estimates exist
  - Very useful for tuning the algorithm to required accuracy
- Not so good for inhomogenous systems
- Overall complexity: $O(N \log N)$
- Parallelization: FFT requires all-to-all communication
- ScaFaCoS:
  - P³M (Stuttgart)
  - NFFT (Chemnitz): 3D FFT Scaling seems to be good

Scaling of the parallel 3D-FFT in ScaFaCoS (Mesh $512^3$, up to 262144 cores)
Multigrid

- Multigrid Poisson Solver (Brandt 1977)
  - Near field as Ewald Summation
  - Far field
    - Hierarchy of successively finer grids
    - Solve Poisson equation on coarsest grid
    - Smooth out high-frequency errors
    - Prolongate correction to finer grid
    - Repeat until finest grid is reached
    - Downsample residual error
- $O(N)$
- Open BC (PBC in development)
- Parallelization: scaling graph
- ScaFaCoS:
  - PP3MG (Wuppertal)
  - VMG (Bonn)
Barnes-Hut Tree Code

- Tree code (Barnes/Hut 1988)
  - Bundle charges in hierarchical tree
  - Multipole development of bundles
  - Propagate multipole terms up the tree
  - Depending on distance in tree of bundle, use multipole terms of higher levels

- Open or closed BC
- \( O(N \log N) \)
- Parallelization: scaling graph
- ScaFaCoS: PEPC (Jülich)
Fast Multipole Method

- FMM (Greengard/Rokhlin 1987)
  - *Like Barnes-Hut Tree code on a grid*
  - *Propagate multipole terms up the hierarchy of coarser grids*
  - *Use multipole terms of coarser grid for larger distances*

- Open BC (PBC in development)

- O(N)

- Parallelization: scaling graph

- ScaFaCoS: FMM (Jülich)
Maxwell Equation Molecular Dynamics (MEMD)

  - In principle, Maxwell's equations are local
  - Problem is the large speed of light
  - Simply assume that the speed of light is small
  - Statistics is still correct!

- Method
  - Simulate the propagation of $E$, $j$, $H$, on a cubic lattice (plaquettes)
  - Use same time steps as MD

- Can handle dielectric variations
- $O(N)$
- Parallelization: Algorithm is fully local!
- ScaFaCoS: MEMD (Stuttgart)

- A. C. Maggs and V. Rosetto, PRL 88:196402, 2002
- I. Pasichnyk and B. Dünweg, JPCM 16:3999, 2004
Summary

- MD poses some challenges for the exascale
- Long-range interactions are a significant problem
- ScaFaCoS will provide the best state-of-the-art algorithms for computing long-range interactions

Acknowledgements

Bonn
Multigrid: M.Griebel, R.Wildenhues

Chemnitz
Parallel sorting: M.Hofmann, G.Rünger
NFFT/PFFT: M.Pippig, D.Potts

Jülich
Project Coordination: G.Sutmann
Interface: R.Halver.
FMM: H.Dachsel, I.Kabadchov
Barnes-Hut: L.Arnold, P.Gibbon, R.Speck
Multigrid: B.Steffen

Stuttgart
MEMD: C. Holm, F.Fahrenberger
P3M, MMM, ELC: A.Arnold, O.Lenz

Wuppertal
Multigrid: M.Bolten
And now for something completely different...
Coarse-grained simulations

ESPResSo

Olaf Lenz
Coarse-graining

- Even with exascale, many systems are out of reach
- Instead of going to higher resolution, go to lower resolution!
- Model only *important* degrees of freedom
- Reduce number of model parameters
- Top-down modelling

Pros
- *Much shorter simulation times*
- *No exascale HPC machines*
- *Parameter space can be sampled much better*
- *Effects of parameters are better understood*

Cons
- *Quantitative results often not as good*
- *Often requires non-standard methods*
Coarse-graining

- Time scale
- Length scale
- Atomistic
- Molecular
- Quantum
- Soft fluid
- Finite elements
- Coarse-graining
ESPResSo

- **Extensible Simulation Package for Research on Soft matter systems**
- Developed originally at Max-Planck-Institute for Polymer Research in Mainz
- Now maintained at ICP Stuttgart
- Created by 3 PhD students during their thesis (Axel Arnold, Bernward Mann and Hanjo Limbach)
- Open-source, GPL
- Intended for *coarse-grained* bead-spring models (with a focus on charged systems)

http://espressomd.org
Why yet another Simulation Package?

- Bead-spring models: Combine several atoms into a single bead
- Often requires uncommon methods
  - Special interactions (DPD, Gay-Berne ellipsoids, ...)
  - Special integrators (MCPD, Hybrid MC/MD, ...)
  - Combined with lattice models (Lattice-Boltzmann, MEMD, ...)
  - Uncommon simulation protocols (Simulated annealing, Parallel tempering, ...)
  - Special constraints (Walls, Pores, ...)
- Standard MD simulation packages (GROMACS, NAMD, AMBER, ...) are not flexible enough to deal these models

Package must be flexible!

- In research, new methods are developed
- Building new methods into highly optimized code is very hard

Package must be extensible!
ESPResSo Overview

- Simulation core written in ANSI C
- MPI parallelized
- Simulation core is controlled by scripting language (currently Tcl, future Python)
- A simulation is defined by an “ESPResSo script”
- Free and Open-Source (GPL)
- Extensible: readability preferred over performance
- But of course not as optimized as e.g. GROMACS
Methods in ESPResSo

- **Integrators and ensembles**: Velocity-Verlet algorithm (NVE), Langevin thermostat (NVT), Barostat by Dünweg (NPT), Quaternion integrator for non-spherical particles or point-like dipoles

- **Nonbonded interactions**: Lennard-Jones, Gay-Berne, Buckingham, Morse, …

- **Bonded interactions**: harmonic, FENE, tabulated, bond-angle interaction, dihedral interaction

- **Long-range interactions**: for electrostatics: P3M, MMM1D, MMM2D, Ewald, ELC and MEMD; for point-like dipoles: dipolar P3M

- **Hydrodynamic interactions**: DPD, Lattice-Boltzmann fluid (on GPGPU) coupled to particle simulation

- **Constraints**: Particles can be fixed in any directions; walls, pores, spheres...

- **Analysis**: energy components, pressure tensor, forces, distribution functions, structure factors, polymer-specific analysis functions (radius of gyration, …), output to VMD

- ...and it is continuously growing...
That's all.
Thanks for your attention!