Computational Rheology via LAMMPS, October 12, 2013
85th Meeting of the Society of Rheology

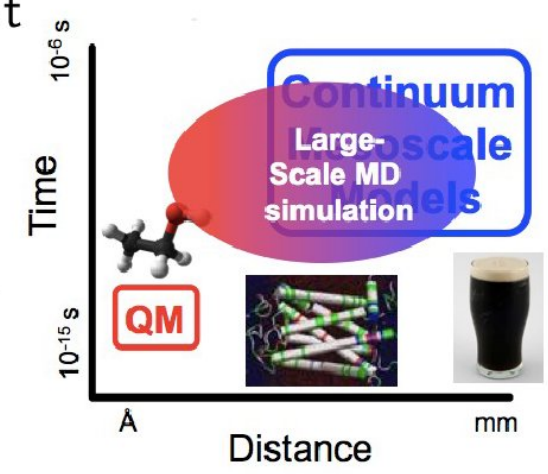
1 – Overview of Molecular/Particulate Dynamics

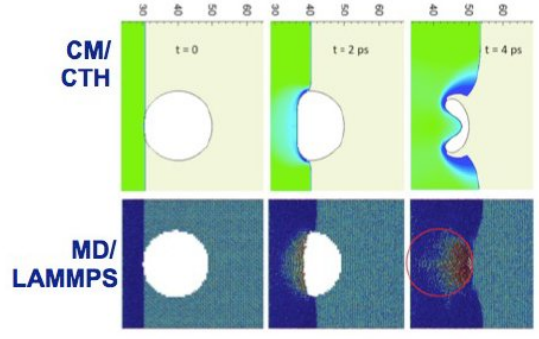
Steve Plimpton
sjplimp@sandia.gov
Computational Science Center
Sandia National Laboratories,
Albuquerque, New Mexico


Sandia National Laboratories is a multi program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

What is classical MD good for?

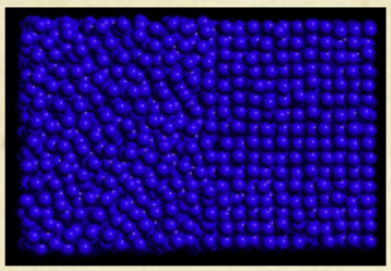
- Quantum mechanics (QM) accurate at atomic scale: ∼1000 atoms
- Atomic-scale phenomena usually play out at much larger scale
- Mesoscale bigger than atoms, smaller than macroscopic: $\sim 10^{23}$ atoms
- QM and CM (continuum/mesoscale) models cannot be directly compared
- Small molecular dynamics (MD) simulations can reproduce QM
- MD can scale up to millions (billions) of atoms, overlapping low-end of CM
- MD can inform CM models with QM-accurate results

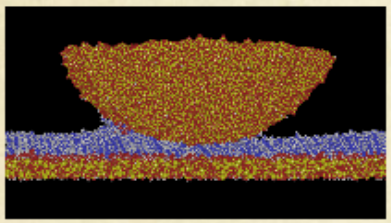




CTH images courtesy of David Damm, Sandia

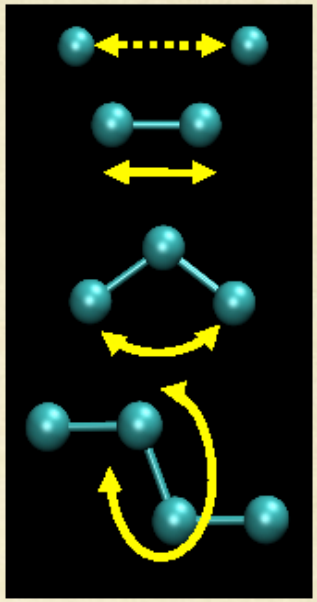
How does classical MD work?





Initial positions and velocities

Interatomic potential



- Classical simulation technique
- Empirical interactions
- Evolve system, analyze

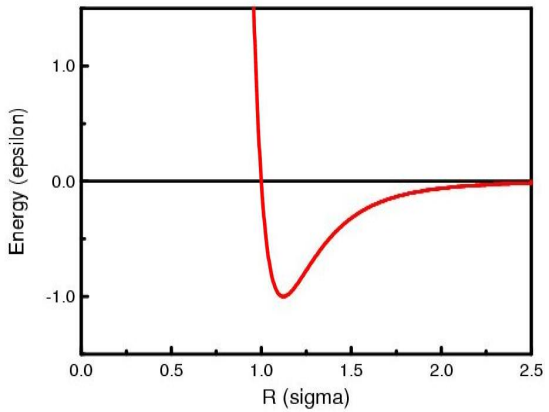
⇨

Positions and velocities at later times

Thanks to Aidan Thompson

Classical MD basics

- Each of N particles is a point mass
  - atom
  - group of atoms (united atom)
  - macro- or meso- particle
- Particles interact via empirical force laws
  - all physics in energy potential $\Rightarrow$ force
  - pair-wise forces (LJ, Coulombic)
  - many-body forces (EAM, Tersoff, REBO)
  - molecular forces (springs, torsions)
  - long-range forces (Ewald)
- Integrate Newton's equations of motion
  - F = ma
  - set of 3N coupled ODEs
  - advance as far in time as possible
- Properties via time-averaging ensemble snapshots (vs MC sampling)



MD timestep

- Velocity-Verlet formulation:
  - update V by 1/2 step (using F)
  - update X (using V)
  - build neighbor lists (occasionally)
  - compute F (using X)
  - apply constraints & boundary conditions (on F)
  - update V by 1/2 step (using new F)
  - output and diagnostics

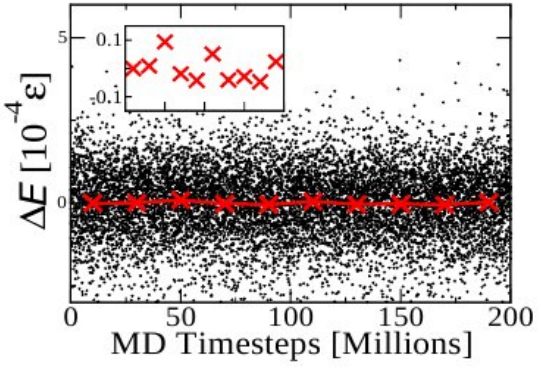
- CPU time break-down:
  - inter-particle forces = 80%
  - neighbor lists = 15%
  - everything else = 5%

Aside on MD integration schemes

Most MD codes use some form of explicit Stormer-Verlet

- Only second-order: $\Delta E = |\langle E \rangle - E_0| \sim \Delta t^2$
- Global stability trumps local accuracy of high-order schemes
- Can be shown that for Hamiltonian equations of motion, Stormer-Verlet exactly conserves a shadow Hamiltonian and $E - E_s \sim O(\Delta t^2)$
- For users: no energy drift over millions of timesteps
- For developers: easy to decouple integration scheme from efficient algorithms for force evaluation, parallelization

32 atom LJ cluster
200M timesteps
$\Delta t = 0.005$



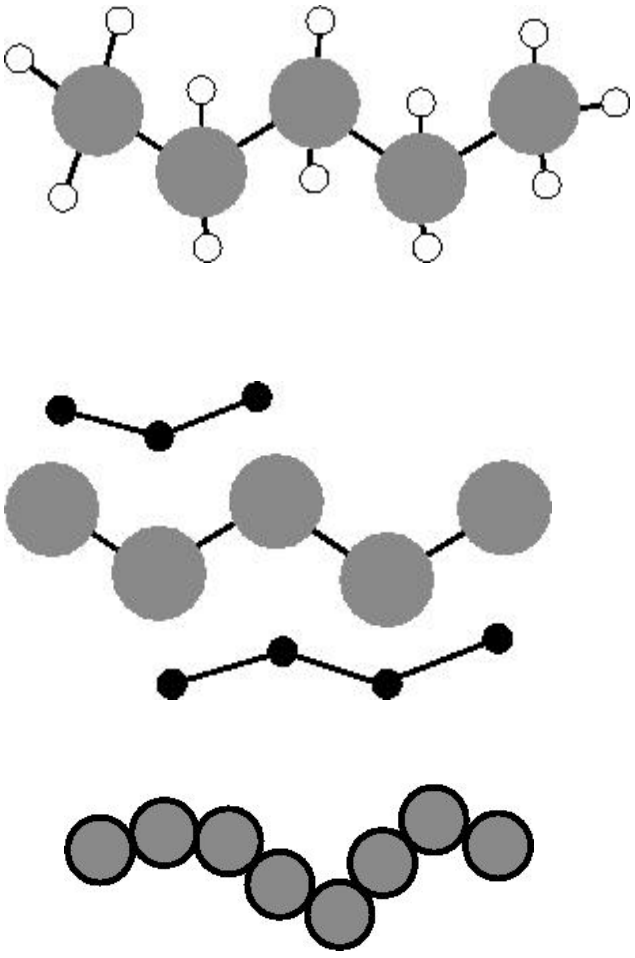
Computational issues

- Are always limited in number of atoms and length of time you can simulate

- These have a large impact on CPU cost of a simulation:
  - level of detail in model
  - cutoff distance of force field
  - long-range Coulombics
  - finding neighbors
  - timestep size
  - parallelism

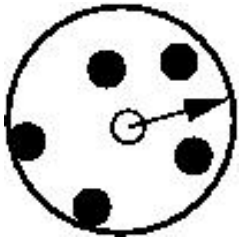
Coarse-graining of polymer models

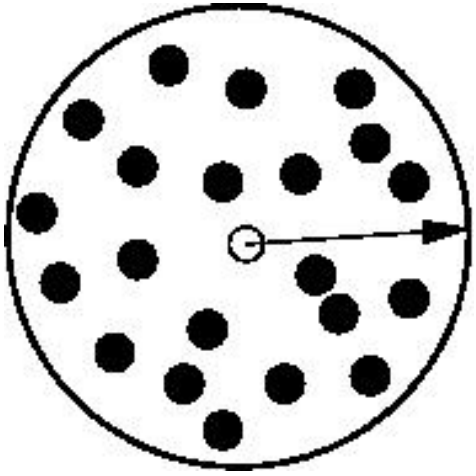
- All-atom:
  - Δt = 0.5-1.0 fmsec for C-H
  - C-C distance = 1.5 Angs
  - cutoff = 10 Angs
- United-atom:
  - # of interactions is 9x less
  - Δt = 1.0-2.0 fmsec for C-C
  - cutoff = 10 Angs
  - 20-30x savings over all-atom
- Bead-Spring:
  - 2-3 C per bead
  - $\Delta t \Longleftrightarrow$ fmsec mapping is T-dependent
  - $2^{1/6}\sigma$ cutoff $\Rightarrow$ 8x in interactions
  - can be considerable savings over united-atom



Cutoff in force field

- Forces = 80% of CPU cost
- Short-range forces:
  - O(N) scaling for classical MD
  - constant density assumption
  - pre-factor is cutoff-dependent
- # of pairs/atom = cubic in cutoff
  - 2x the cutoff $\Rightarrow$ 8x the work
- Use as short a cutoff as can justify:
  - LJ = 2.5σ (standard)
  - all-atom and UA = 8-12 Angstroms
  - bead-spring = $2^{1/6}\sigma$ (repulsive only)
  - Coulombics = 12-20 Angstroms
  - solid-state (metals) = few neighbor shells due to screening
- Test sensitivity of your results to cutoff



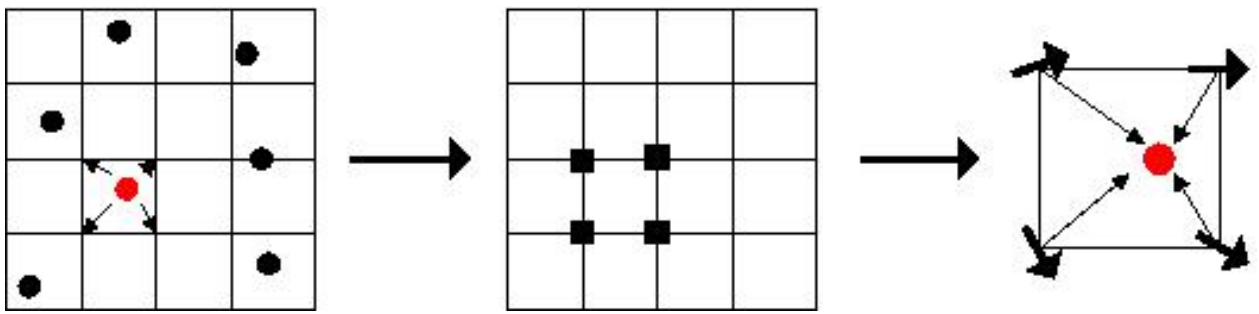


Long-range Coulombics

- Systems that need it:
  - charged polymers (polyelectrolytes)
  - organic & biological molecules
  - ionic solids, oxides
  - not most metals (screening)
- Computational issue:
  - Coulomb energy only falls off as $1/r$
- Options:
  - cutoff: scales as N, but large contribution at 10 Angs
  - Ewald: scales as $N^{3/2}$
  - particle-mesh Ewald: scales as $N \log(N)$
  - multipole: scales as N, but doesn't beat PME
  - multi-level summation: scales as N
    can beat PME for low-accuracy, large proc count

PPPM (Particle-mesh Ewald)

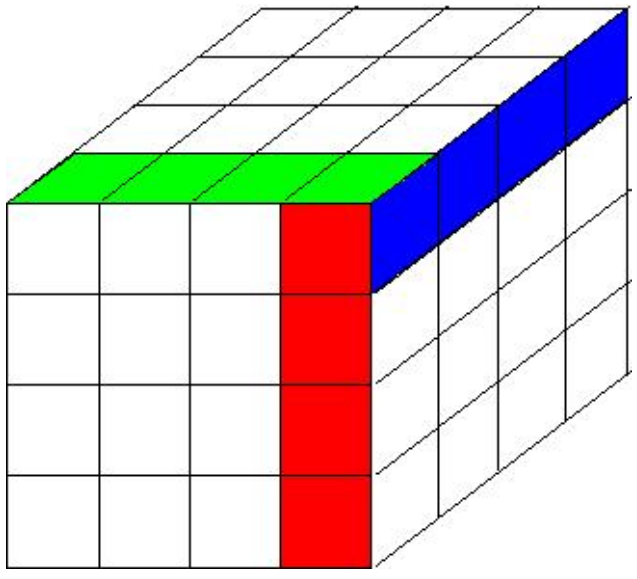
- *Hockney & Eastwood, Comp Sim Using Particles (1988).*
- *Darden, et al, J Chem Phys, 98, p 10089 (1993).*
- Like Ewald, except sum over periodic images evaluated:
  - interpolate atomic charge to 3d mesh
  - solve Poisson's equation on mesh (4 FFTs)
  - interpolate E-fields back to atoms



- User-specified accuracy + cutoff $\Rightarrow$ ewald-G + mesh-size
- Scales as $N\sqrt{\log(N)}$ if grow cutoff with N
- Scales as $N\log(N)$ if cutoff held fixed

Parallel FFTs (in LAMMPS)

- 3d FFT is 3 sets of 1d FFTs
  - in parallel, 3d grid is distributed across procs
  - 1d FFTs on-processor
  - native library or FFTW (www.fftw.org)
  - multiple transposes of 3d grid
  - data transfer can be costly
- FFTs for PPPM can scale poorly
  - on large # of procs and on clusters

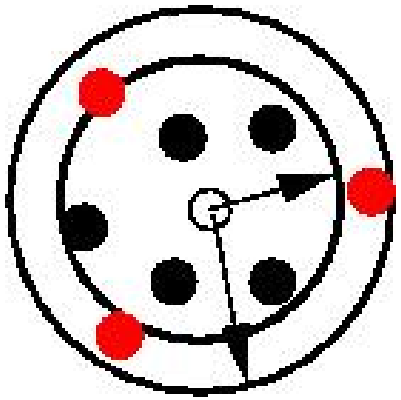


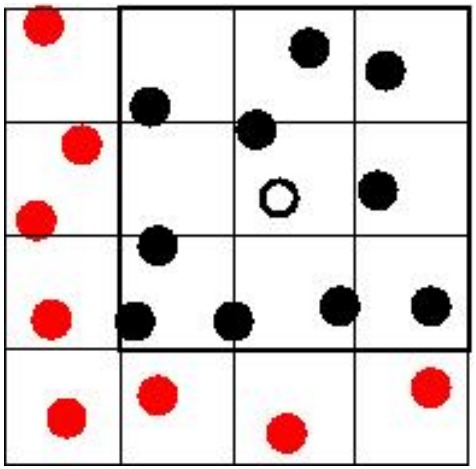
Good news: Cost of PPPM is only ∼2x more than 8-10 Ang cutoff

Neighbor lists

Problem: how to efficiently find neighbors within cutoff?

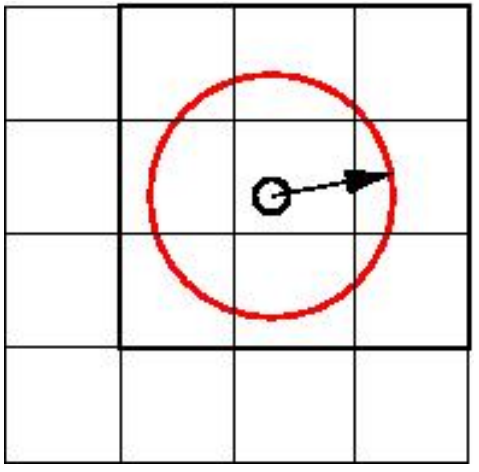
- For each atom, test against all others
  - $O(N^2)$ algorithm
- Verlet lists:
  - *Verlet, Phys Rev, 159, p 98 (1967)*
  - $R_{neigh} = R_{force} + \Delta_{skin}$
  - build list: once every few timesteps
  - other timesteps: scan larger list for neighbors within force cutoff
  - rebuild: any atom moves 1/2 skin
- Link-cells (bins):
  - *Hockney et al, J Comp Phys, 14, p 148 (1974)*
  - grid domain: bins of size R_{force}
  - each step: search 27 bins for neighbors (or 14 bins)





Neighbor lists (continued)

- Verlet list is ~6x savings over bins
  - $V_{sphere} = 4/3\ \pi r^3$
  - $V_{cube} = 27\ r^3$



- Fastest methods do both
  - link-cell to build Verlet list
  - use Verlet list on non-build timesteps
  - $O(N)$ in CPU and memory
  - constant-density assumption
  - this is what LAMMPS implements

Timescale in classical MD

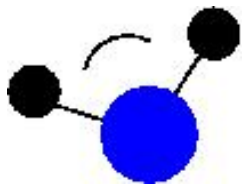
- Timescale of simulation is most serious bottleneck in MD
- Timestep size limited by atomic oscillations
  - C-H bond = 10 fmsec $\Rightarrow$ 1/2 to 1 fmsec timestep
  - Debye frequency = 10^{13} $\Rightarrow$ 2 fmsec timestep
- Reality is often on a much longer timescale
  - protein folding (msec to seconds)
  - polymer entanglement (msec and up)
  - glass relaxation (seconds to decades)
  - rheological experiments (Hz to KHz)
- Even smaller timestep for tight-binding or quantum-MD

Particle-time metric

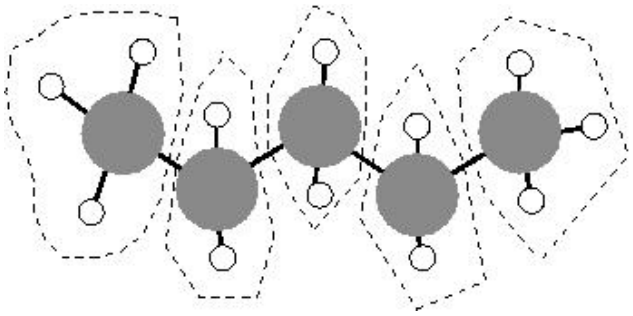
- Atom * steps = size of your simulation
- Up to 10^{12} is desktop scale $\Rightarrow$ 10^6 atoms for 10^6 timesteps
  - 1 μsec/atom/step on CPU core (cheap LJ potential)
  - 2 weeks on single core, 1 day on multi-core desktop
- 10^{12} to 10^{14} is cluster scale
- 10^{14} and up is supercomputer scale
- 1 cubic micron (10^{10} atoms) for 1-2 nanoseconds (10^6 steps)
  - 1000 flops per atom per step $\Rightarrow$ 10^{19} flops
  - MD is 10% of peak $\Rightarrow$ 1 day on a Petaflop machine
- GPUs are changing landscape:
  can be 5-10x faster than multicore CPU

Extending timescale via SHAKE

- *Ryckaert, et al, J Comp Phys, 23, p 327 (1977)*
- Add constraint forces to freeze bond lengths & angles
  - rigid water (TIP3P)
  - C-H bonds in polymer or protein



- Extra work to enforce constraints:
  - solve matrix for each set of non-interacting constraints
  - matrix size = # of constraints
- Allows for 2-3 fmsec timestep



Extending timescale via rRESPA

- *Tuckerman et al, J Chem Phys, 97, p 1990 (1992)*
- reversible REference System Propagator Algorithm
- Rigorous multiple timestep method
  - time-reversible
  - operator calculus $\Rightarrow$
    derivation of conserved ensemble quantities
- Sub-cycle on fast degrees of freedom
  - innermost loop on bond forces (0.5 fmsec)
  - next loop on 3-4 body forces
  - next loop on van der Waals & short-range Coulombic
  - outermost loop on long-range Coulombic (4 fmsec)
- Can yield 2-3x speed-up, less in parallel due to communication

Classical MD in parallel

- MD is inherently parallel
  - forces on each atom can be computed simultaneously
  - X and V can be updated simultaneously
- Nearly all MD codes are parallelized
  - distributed-memory message-passing (MPI) between nodes
  - MPI or threads (OpenMP, GPU) within node
- MPI = message-passing interface
  - MPICH or OpenMPI
  - assembly-language of parallel computing
  - lowest-common denominator
  - most portable
  - runs on all parallel machines, even on multi- and many-core
  - more scalable than shared-memory parallel

Goals for parallel algorithms

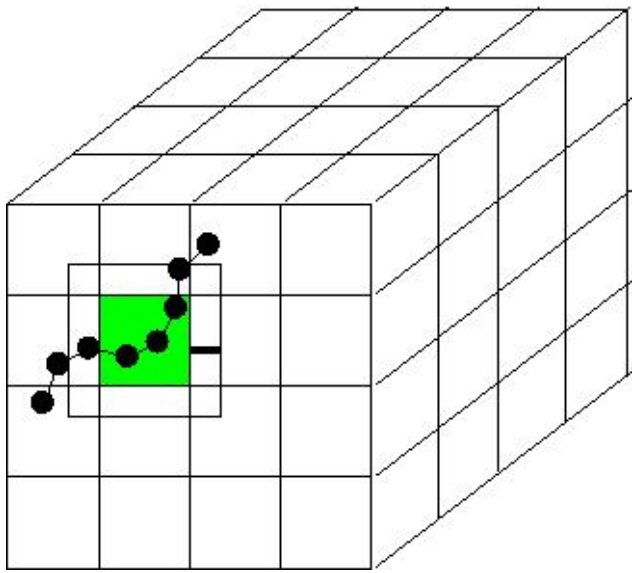
- Scalable
  - short-range MD scales as N
  - optimal parallel scaling is N/P
  - even on clusters with higher communication costs
- Good for short-range forces
  - 80% of CPU
  - long-range Coulombics have short-range component
- Fast for small systems, not just large
  - nano, polymer, bio systems require long timescales
  - 1M steps of 10K atoms is more useful than 10K steps of 1M atoms
- Efficient at finding neighbors
  - liquid state, polymer melts, small-molecule diffusion
  - neighbors change rapidly
  - atoms on a fixed lattice is simpler to parallelize

Parallel algorithms for MD

- *Plimpton, J Comp Phys, 117, p 1 (1995)*
- 3 classes of algorithms used by all MD codes
  1. atom-decomposition = split and replicate atoms
  2. force-decomposition = partition forces
  3. spatial-decomposition = geometric split of simulation box
- All 3 methods balance computation optimally as N/P
- Differ in key issues for parallel scalability
  - communication costs
  - load-balance
- Focus on inter-particle force computation,
  other tasks can be done within any of 3 algorithms
  - molecular forces
  - time integration (NVE/NVT/NPT)
  - thermodynamics, diagnostics, ...

Spatial-decomposition algorithm

- Physical domain divided into 3d boxes, one (or more) per processor
- Each proc computes forces on atoms in its box using ghost info from nearby processors
- Atoms carry along molecular topology as they migrate to new procs
- Communication via 6-way stencil
- Advantages
  - communication scales sub-linear as $(N/P)^{2/3}$, for large problems
  - memory is optimal N/P
- Disadvantages
  - more complex to code efficiently
  - load-imbalance can be problematic



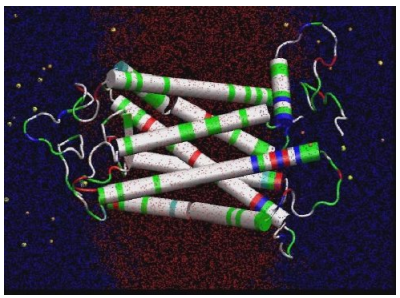
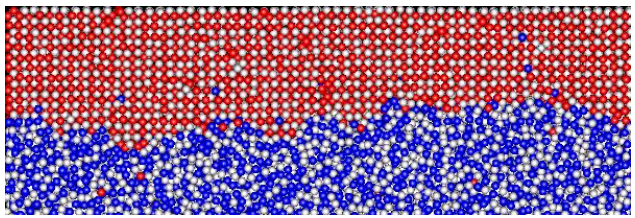
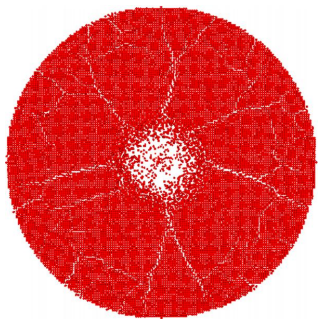
Freely available parallel MD codes

- Bio-oriented MD codes
  - CHARMM: original protein force fields
  - AMBER: original DNA force fields
  - NAMD: fast and scalable
  - Gromacs: fastest and scalable
- Materials-oriented MD codes (can also do bio problems):
  - DL_POLY: distributed by Daresbury Lab, UK
  - LAMMPS: distributed by Sandia National Labs
- GPU-centric MD code (materials and bio):
  - HOOMD: distributed by U Michigan
  - codes above have GPU-capable kernels

What is LAMMPS?

Large-scale Atomic/Molecular Massively Parallel Simulator
http://lammps.sandia.gov

- Classical MD code
- Open source (GPL), highly portable C++
- 3-legged stool: bio, materials, mesoscale

- Particle simulator at varying length and time scales
  electrons $\Rightarrow$ atomistic $\Rightarrow$ coarse-grained $\Rightarrow$ continuum
- Spatial-decomposition of simulation domain for parallelism
- Energy minimization, dynamics, non-equilibrium MD
- GPU and OpenMP enhanced
- Can be coupled to other scales: QM, kMC, FE, CFD, ...

Reasons to use LAMMPS

1. Versatile
   - bio, materials, mesoscale
     - Sat AM: Tour of LAMMPS Features
   - atomistic, coarse-grained, continuum
     - Sat PM: Coarse-grain Applications with LAMMPS
2. Good parallel performance
   - Sat AM: Tour of LAMMPS Features
3. Easy to extend
   - Sun PM: Modifying LAMMPS and New Developments
4. Well documented
   - extensive web site
   - 1200 page manual
5. Active and supportive user community
   - 40K postings to mail list, 1200 subscribers
   - quick turn-around on Qs posted to mail list

Another reason to use LAMMPS

6. Features for rheology (next 2 days)
   - Mesoscale models:
     - DPD = dissipative particle dynamics
     - SPH = smoothed particle hydrodynamics
     - granular = normal & tangential friction
     - FLD = fast lubrication dynamics
     - PD = peridynamics
     - rigid body dynamics
   - Aspherical particles
     - point ellipsoids
     - rigid body collections of points, spheriods, ellipsoids
     - rigid bodies of triangles (3d) and lines (2d)
   - Coarse-grained solvent models
     - rigid water
     - polymers (united-atom, bead-spring)
     - LJ particles
     - stochastic rotation dynamics (SRD)
     - implicit

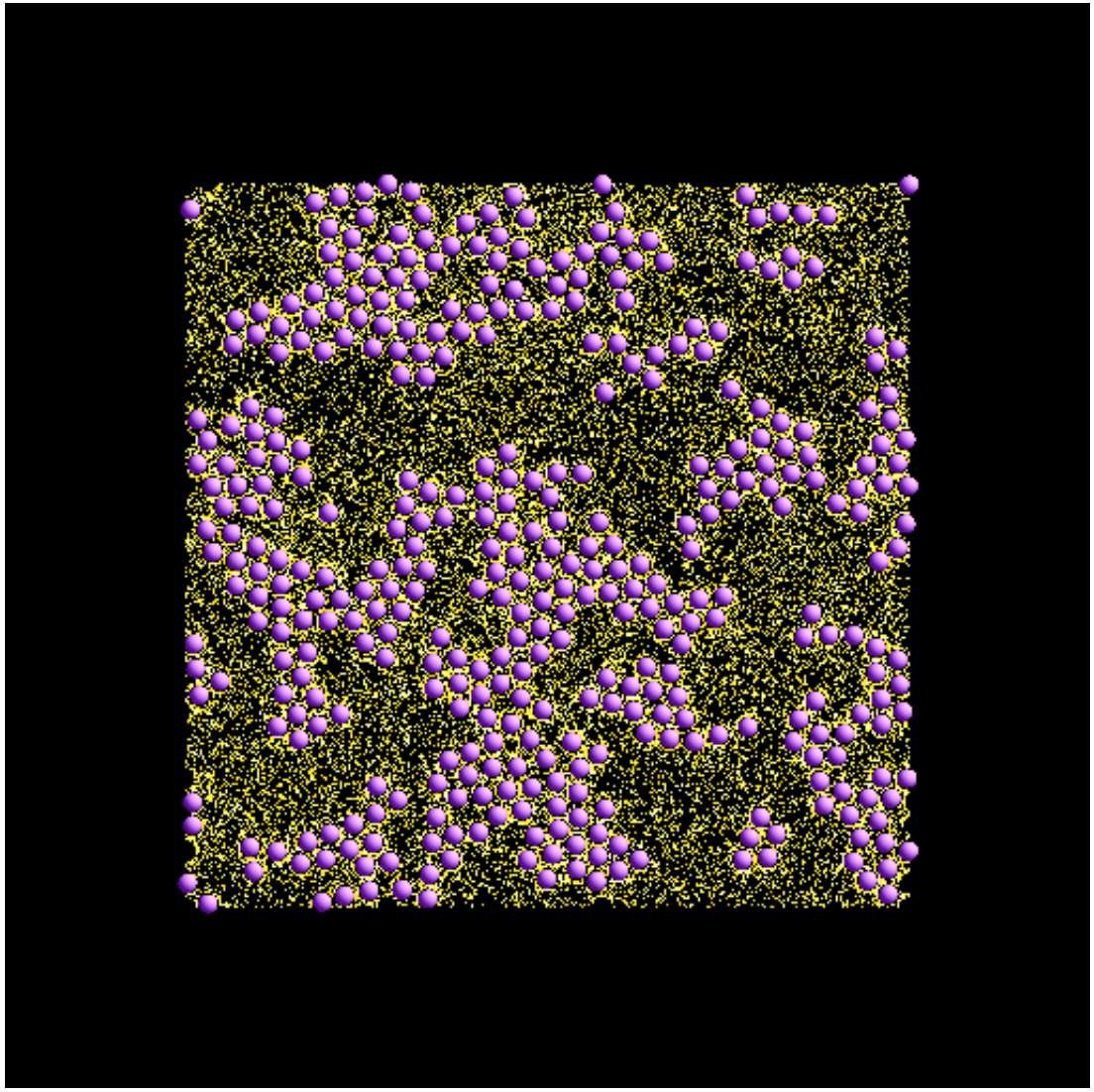
More rheological options in LAMMPS

Many of these options came from 4-year collaboration with 3M, BASF, Corning, P&G on solvated colloidal modeling

- Particle/particle interactions:
  - pair gayberne, resquared, colloid, yukawa/colloid, vincent
  - pair brownian, lubricate, lubricateU (implicit)
  - pair gran/hooke and gran/hertz
  - pair hybrid/overlay for DLVO models
  - fix srd for colloids + SRD fluid
- Packages:
  - ASPHERE, COLLOID, FLD, GRANULAR
  - RIGID, SRD, USER-LB
- 2 methods for measuring diffusivity
  - mean-squared displacement via compute msd
  - VACF via post-processing of dump file
- 3 methods for measuring shear (or bulk) viscosities
  - NEMD via fix deform and fix nvt/sllod or fix wall
  - Muller-Plathe via fix viscosity
  - Green-Kubo via fix ave/correlate

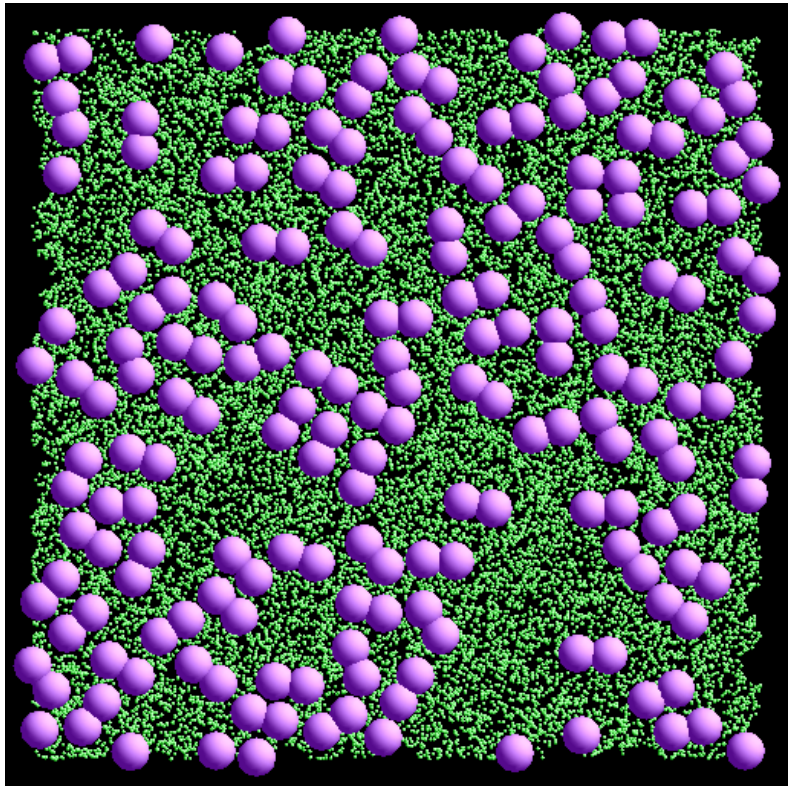
Examples of rheological simulations

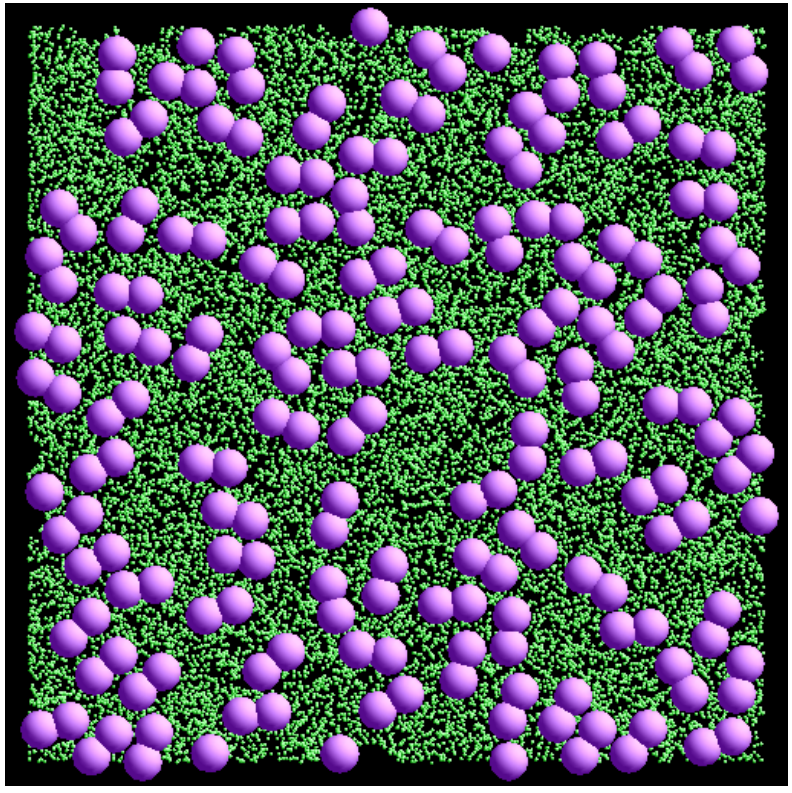
Polymer aggregation under shear



More examples of rheological simulations

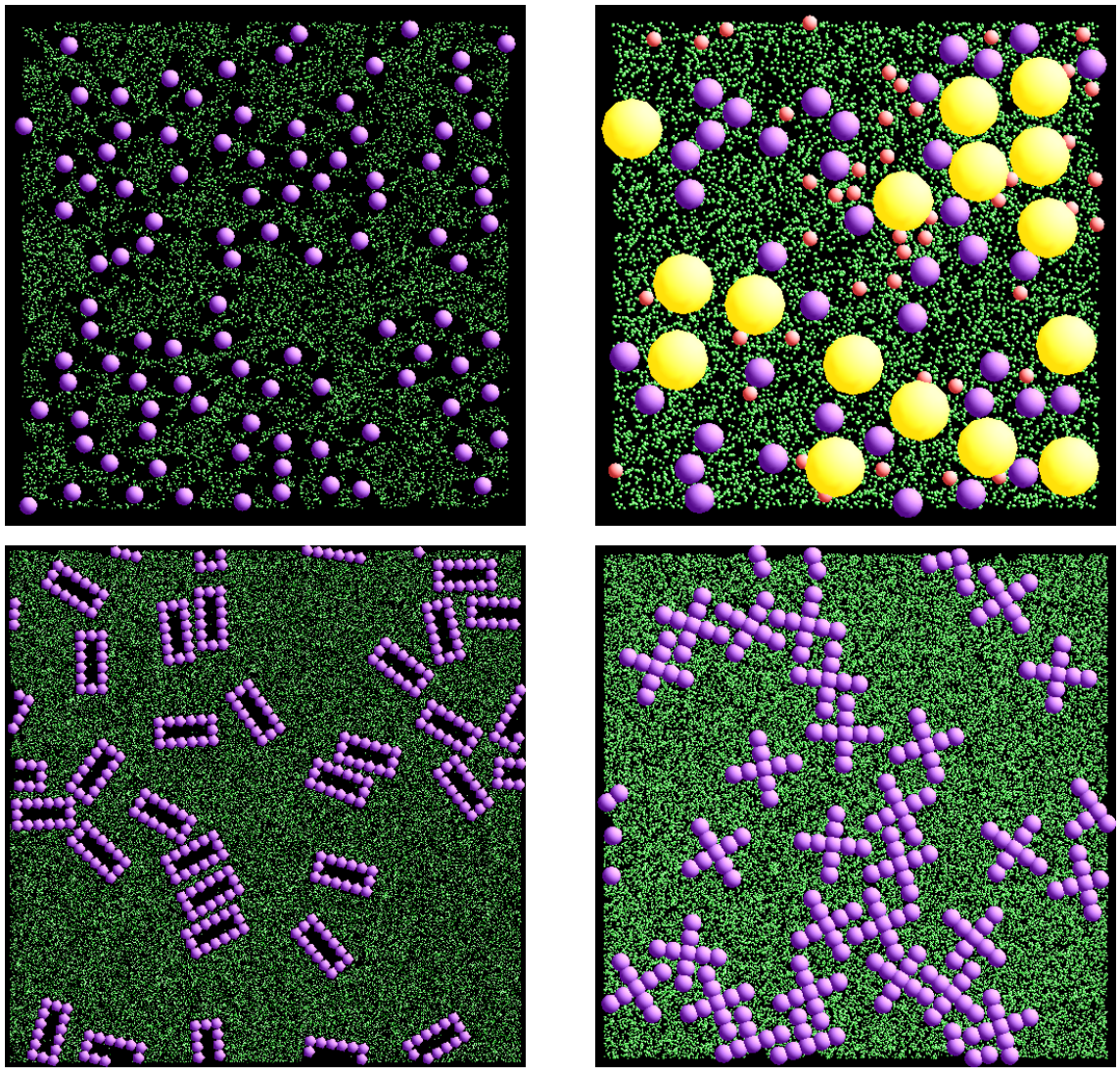
Diffusion and viscosity of solvated dimers





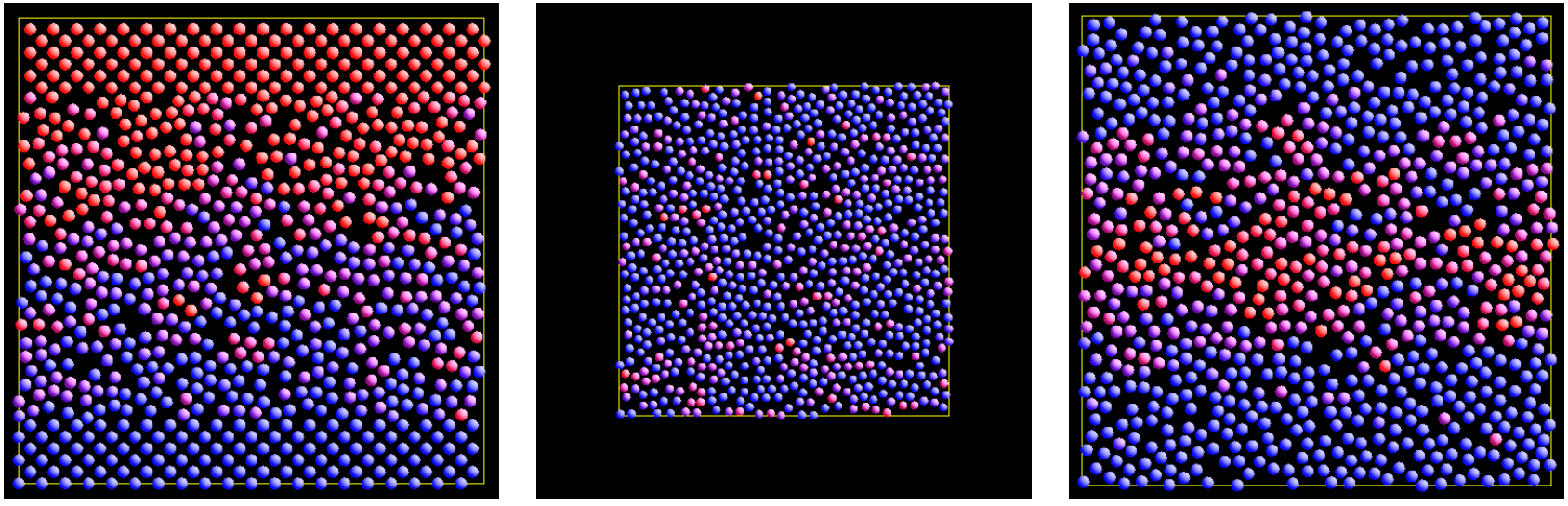
Still more examples of rheological simulations

Viscosity of aspherical in SRD fluid



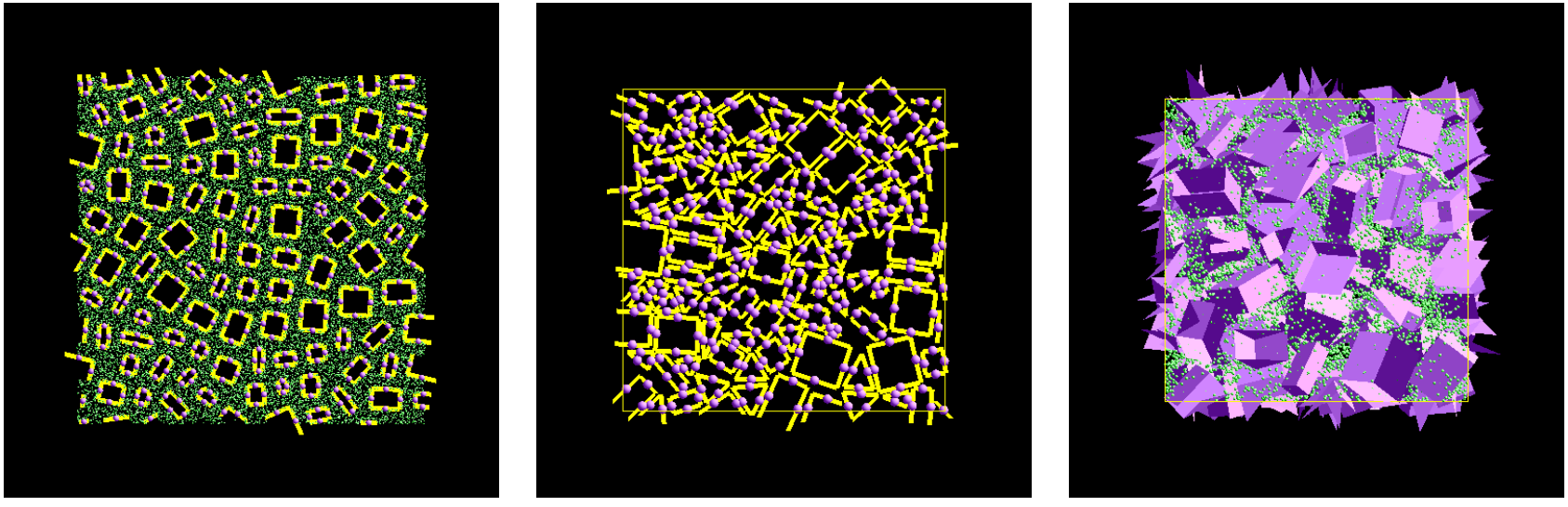
Yet some more examples of rheological simulations

3 methods of measuring viscosity



Finally, enough of rheological simulations

Arbitrary-shape asphericals via lines and triangles



See http://lammps.sandia.gov/movies.html to view all these animations and for links to input scripts