

# What is Classical Molecular Dynamics?

- ▶ Simulation of explicit particles (atoms, ions, ...)
- ▶ Particles interact via relatively simple analytical potential functions
- ▶ Newton's equations of motion are integrated for all particles simultaneously
- ▶ 100-1,000,000's of particles depending on model
- ▶ Time 10 ps to 1  $\mu$ s depending on model (typically ns)

# Why Classical Molecular Dynamics?

- ▶ Includes temperature and pressure effects
- ▶ Able to treat comparatively large systems for a relatively long time
- ▶ Help interpret experiments, and to provide alternative interpretations
- ▶ Test theories of intra- and intermolecular interaction
- ▶ Obtain detailed molecular level information
- ▶ Structure *and* dynamics (compared to QC, MM, MC ...)
- ▶ Improve our understanding of nature through model-building (!!)

# Phase Space and Time Averages

- ▶ A system containing  $N$  atoms has  $6N$  values defining the state of the system ( $3N$  coordinates ( $\mathbf{r}$ ),  $3N$  momenta ( $\mathbf{p}$ ))
- ▶ Each combination of coordinates and momenta define a point in the  $6N$ -dimensional *phase space*:  $\Gamma_N$
- ▶ Consider a property that can be described as a function of the coordinates and momenta,  $A(\Gamma)$ , (for instance the total energy or the (instantaneous) pressure)
- ▶ The average value of the property can then be written as:

$$A_{\text{obs}} = \langle A \rangle_{\text{time}} = \langle A(\Gamma(t)) \rangle_{\text{time}} = \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_{t=0}^{\tau} A(\Gamma(t)) dt$$

- ▶ The *ergodic hypothesis*: “The long time average is equal to the ensemble average in the limit of an infinite number of members in the ensemble”

# Molecular Dynamics

- ▶ A MD simulation generates a sequence of points in phase space connected in time
- ▶ The result is a *trajectory* of all particles in the system as a function of time
- ▶ Time averages and other properties can be calculated from this trajectory
- ▶ Motion of the system through phase space is governed by Hamiltonian equations of motion :

$$\dot{\mathbf{r}}_i = \frac{\partial H}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m_i}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial H}{\partial \mathbf{r}_i} = \mathbf{f}_i$$

- ▶ The Hamiltonian :

$$H(\mathbf{r}, \mathbf{p}) = K(\mathbf{p}) + V(\mathbf{r})$$

$$K(\mathbf{p}) = \sum_i^N \frac{\mathbf{p}_i^2}{2m_i}$$

- ▶  $V$  is a potential energy function
- ▶ Newtons equations of motion :

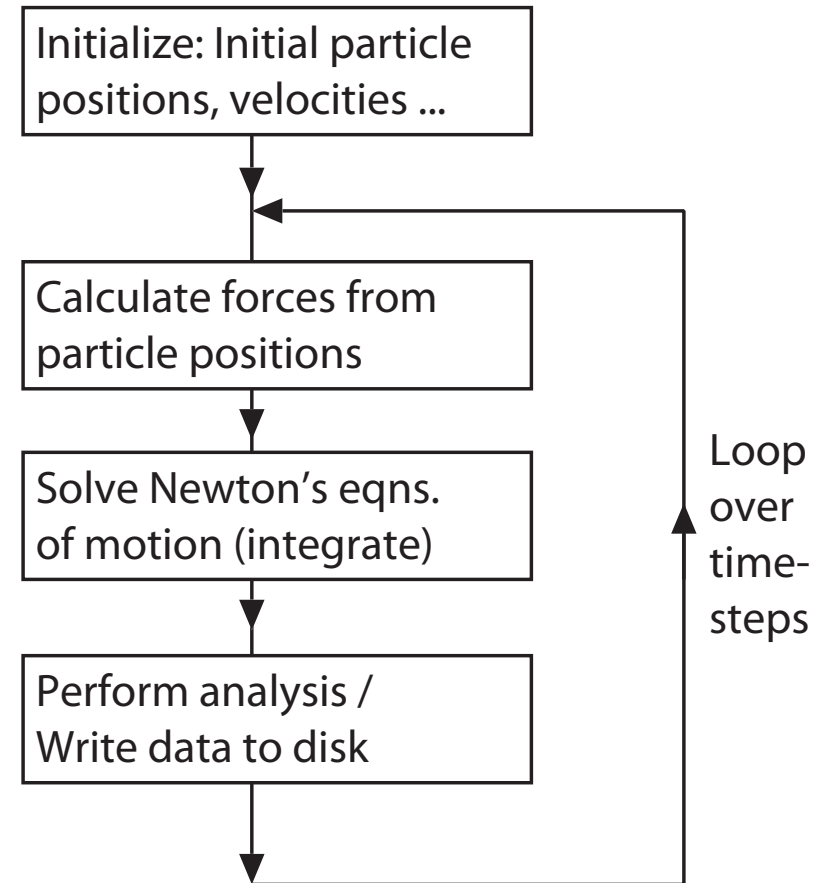
$$\dot{x} = \frac{\partial H}{\partial p_x} = \frac{p_x}{m} = v_x$$

$$\dot{p}_x = -\frac{\partial H}{\partial x} = -\frac{\partial V(x)}{\partial x} = F = ma_x$$

- ▶ The Born-Oppenheimer approximation :
  - ▶ Electrons move much faster than the nuclei
  - ▶ QM: Solve Schrodinger equation for static nuclei
  - ▶ MD: Nuclei experience electrons as an average field

# Steps in Performing an MD Simulation

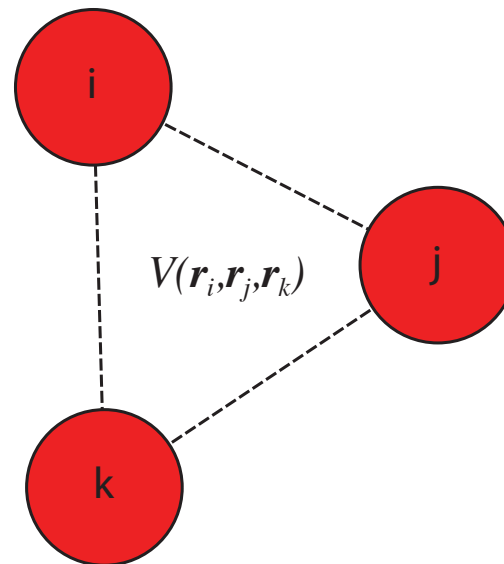
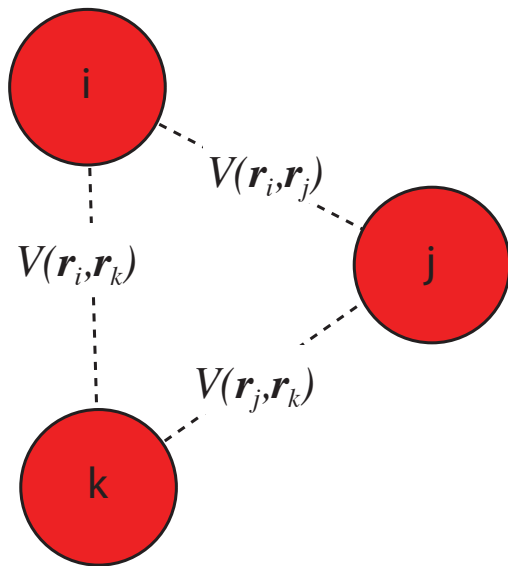
- ▶ Selection of interaction model
- ▶ Selection of boundary conditions
- ▶ Selection of initial conditions (positions, velocities ...)
- ▶ Selection of ensemble (NVE, NVT, NPT ...)
- ▶ Selection of target temperature, density/pressure ...
- ▶ Selection of integrator, thermostat, barostat ...
- ▶ Perform simulation until equilibration is reached (property dependent)
- ▶ Perform production simulation to collect thermodynamic averages, positions, velocities
- ▶ Analyze the results via post-processing



# The Interaction Model

- ▶ The potential energy can be divided into interactions between pairs, triplets, ... of particles :

$$V(\mathbf{r}) = \sum_{ij} V_{ij}(\mathbf{r}_i, \mathbf{r}_j) + \sum_{ijk} V_{ijk}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \sum_{ijkl} V_{ijkl}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l) \dots$$



- ▶ The interaction energy is (usually) largely dominated by the pair term. Truncate the series and make the assumption of *pairwise additivity* :

$$V(\mathbf{r}) \approx \sum_{ij} V_{ij}(\mathbf{r}_i, \mathbf{r}_j)$$

- ▶ In simulations the “true” pair potential is usually replaced by an *effective pair potential*, which incorporate the average many-body effects into the first term :

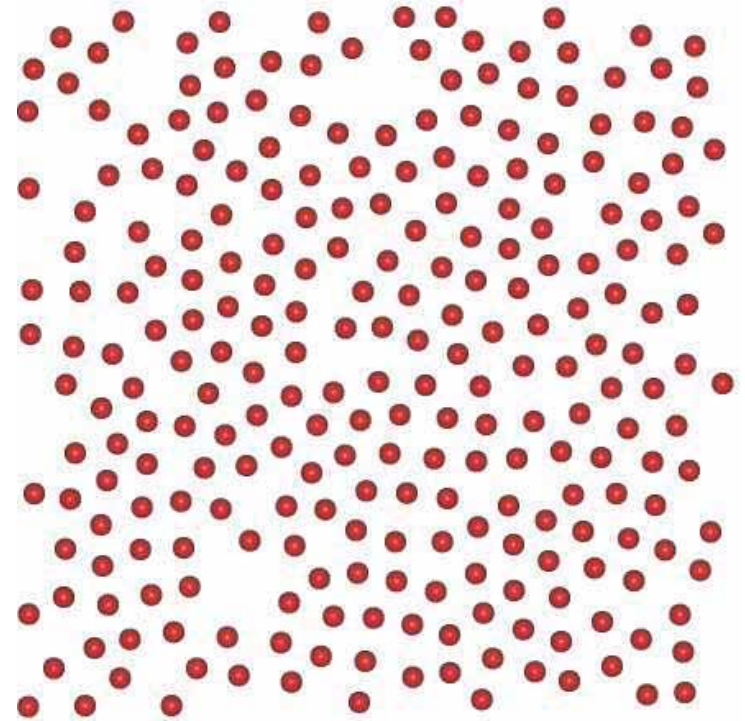
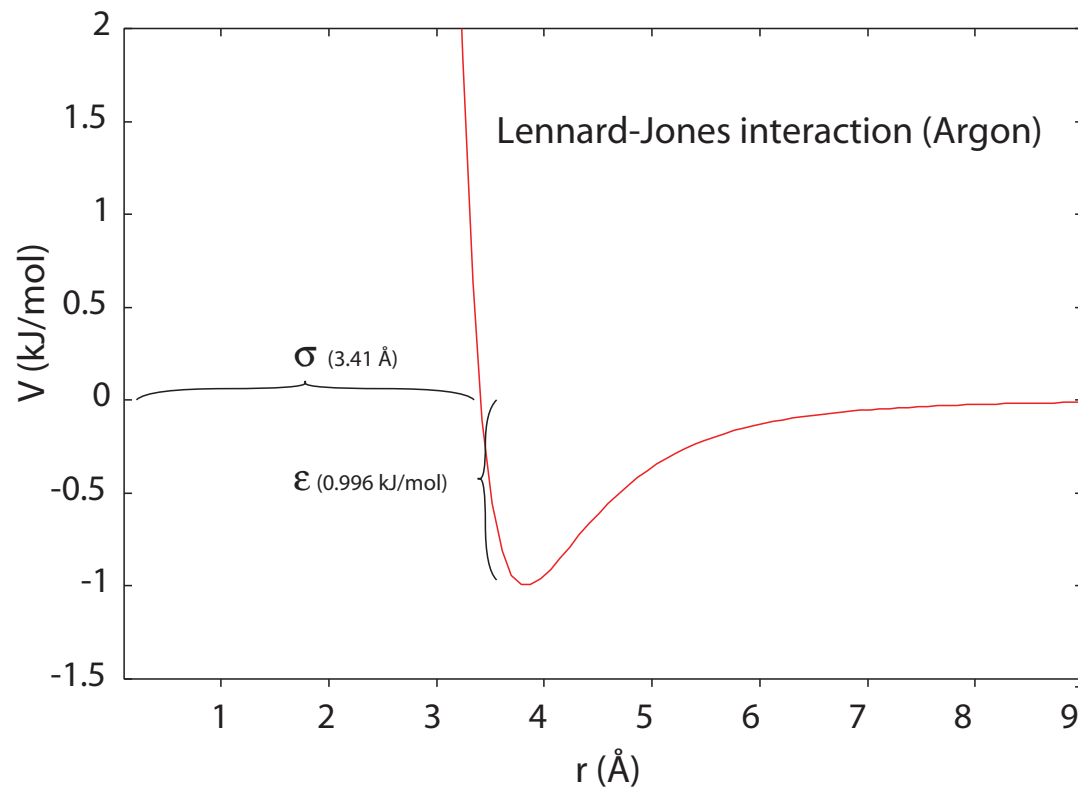
$$V(\mathbf{r}) \approx \sum_{ij} V_{ij}^{\text{eff}}(\mathbf{r}_i, \mathbf{r}_j)$$

- ▶ These potentials are usually parameterized to reproduce some experimental data

# The Lennard-Jones Potential

$$V_{ij}(r_{ij}) = 4\epsilon_{ij} \left( \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right)$$

$$r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$$



Two dimensional simulation of argon

# The Initial Configuration

- ▶ For a solid it may be possible to start from the experimentally determined structure
- ▶ For a liquid it is also common to start from a “solid” lattice, and letting the structure melt
- ▶ For a liquid it is also possible to start from a “random” placement of molecules (this may be tricky)
- ▶ If previous simulation data is available, one might modify a configuration to suit the new simulation, for instance by replacing one or a few particles
- ▶ Macromolecule in liquid: Use experimental coordinates of macromolecule from solid phase structure determination, surround it by liquid

# The Initial Velocities

- ▶ The velocities of the particles is related to the kinetic energy :

$$K = \sum_i^N \frac{m_i v_i^2}{2}$$

- ▶ The kinetic energy is related to the temperature :

$$2 \langle K \rangle = g k_B T = 3 N k_B T$$

- ▶ Assign velocities according to a Boltzmann distribution :

$$P(v_i) = \left( \frac{m_i}{2\pi k_B T} \right)^{1/2} e^{-\frac{m_i v_i^2}{k_B T}}$$

- ▶ Alternatively assign particle velocities from a uniform distribution. They will obtain a Boltzmann distribution after a short time.

- ▶ It is often necessary to adjust the initial velocities so the total linear momentum of the system is zero :

$$\mathbf{p} = \sum_i^N \mathbf{p}_i = \sum_i^N m_i \mathbf{v}_i = \mathbf{0}$$

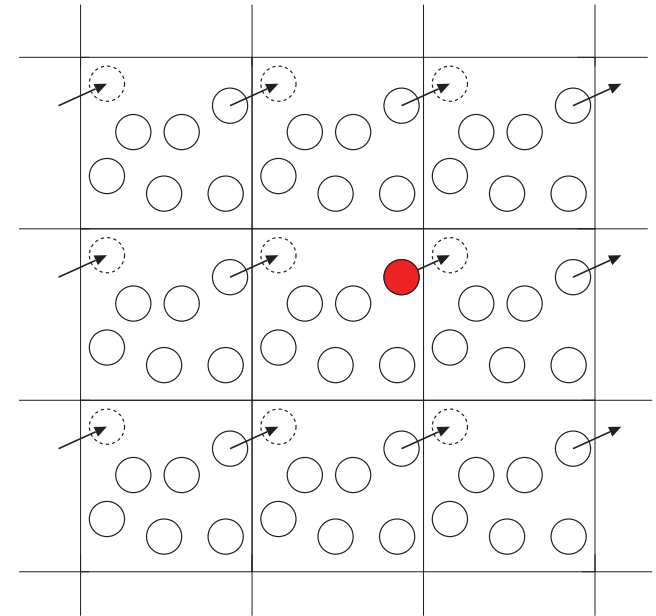
- ▶ This momentum corresponds to translation of the *whole system*
- ▶ Having  $\mathbf{p} \neq \mathbf{0}$  in simulations of “liquid” water, has been referred to as a “flying ice cube”
- ▶ Newton’s equations of motion will preserve the *linear momentum* resulting in a NVE  $\mathbf{P}$  ensemble, resulting in a system with three fewer degrees of freedom
- ▶ For a system with periodic boundary conditions, the *angular momentum* will not be conserved

# Statistical Mechanical Ensembles

- ▶ The “natural” ensemble for MD is the microcanonical ensemble: NVE (constant number of particles, constant volume, constant energy)
- ▶ Typical experiments: Constant pressure and/or temperature
- ▶ Canonical ensemble: NVT (constant number of particles, volume, temperature); Isobaric, isothermal (Gibbs): NPT (constant number of particles, pressure, temperature)
- ▶ Some ways to include such effects in MD
  - ▶ Stochastic events (change kinetic energy of single particle)
  - ▶ Periodic rescaling of velocities ( $2K = gk_B T$ )
  - ▶ Ad hoc partial rescaling of velocities (Berendsen’s thermal bath)
  - ▶ Extended system methods: Include extra variables in the equations of motion such as the volume of the system

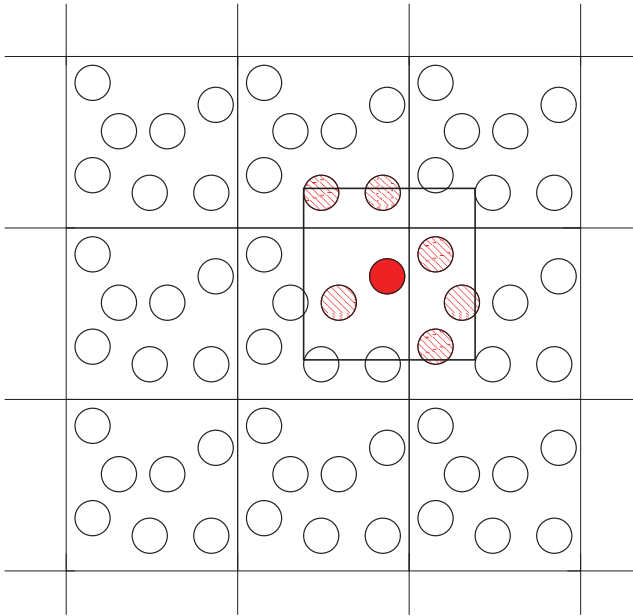
# Periodic Boundary Conditions

- ▶ A “real” system might contain about  $N_A = 6 \cdot 10^{23}$  molecules
- ▶ Simulation may contain about 100-1,000,000 molecules
- ▶ A small cube of 10x10x10 molecules has about half of the molecules on the surface
- ▶ Not appropriate to study “bulk” properties
- ▶ Periodic boundary conditions:
  - ▶ Small box replicated in all directions
  - ▶ A particle that leaves the box on one side is replaced by an image particle that enters from the other side
  - ▶ There are no walls and no surface particles
- ▶ Some systems inherently contains a boundary, for instance a liquid droplet, or a surface



# The Minimum Image Convention

- ▶ Particles interact only with the closest periodic image of the other particles:



How to compute minimum image :

- ▶ Simple and straightforward

```
rx = xi - xj ;  
if ( rx > 0.5 * length )  
    rx = rx - length ;  
if ( rx < -0.5 * length )  
    rx = rx + length ;
```

- ▶ Fortran has a function , `anint`, which returns the nearest integer

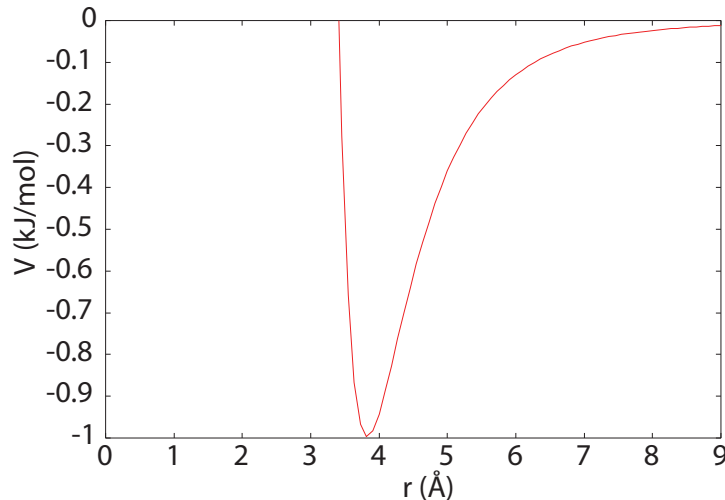
```
rx = rx - length * anint ( rx / length )
```

- ▶ In C, the `floor` function can be used

```
rx = rx - length * floor ( rx / length + 0.5 ) ;
```

# Truncating the Potential

- ▶ The most time consuming part: Computing the nonbonded (vdw & electrostatic) energies and forces
- ▶ For a pair additive potential there are  $N^2$  such interactions
- ▶ Consider the Lennard-Jones argon potential again:



- ▶ Spherical nonbonded cutoff

$$V'_{ij}(r_{ij}) = V_{ij}(r_{ij}) \quad r_{ij} \leq r_c$$
$$V'_{ij}(r_{ij}) = 0 \quad r_{ij} > r_c$$

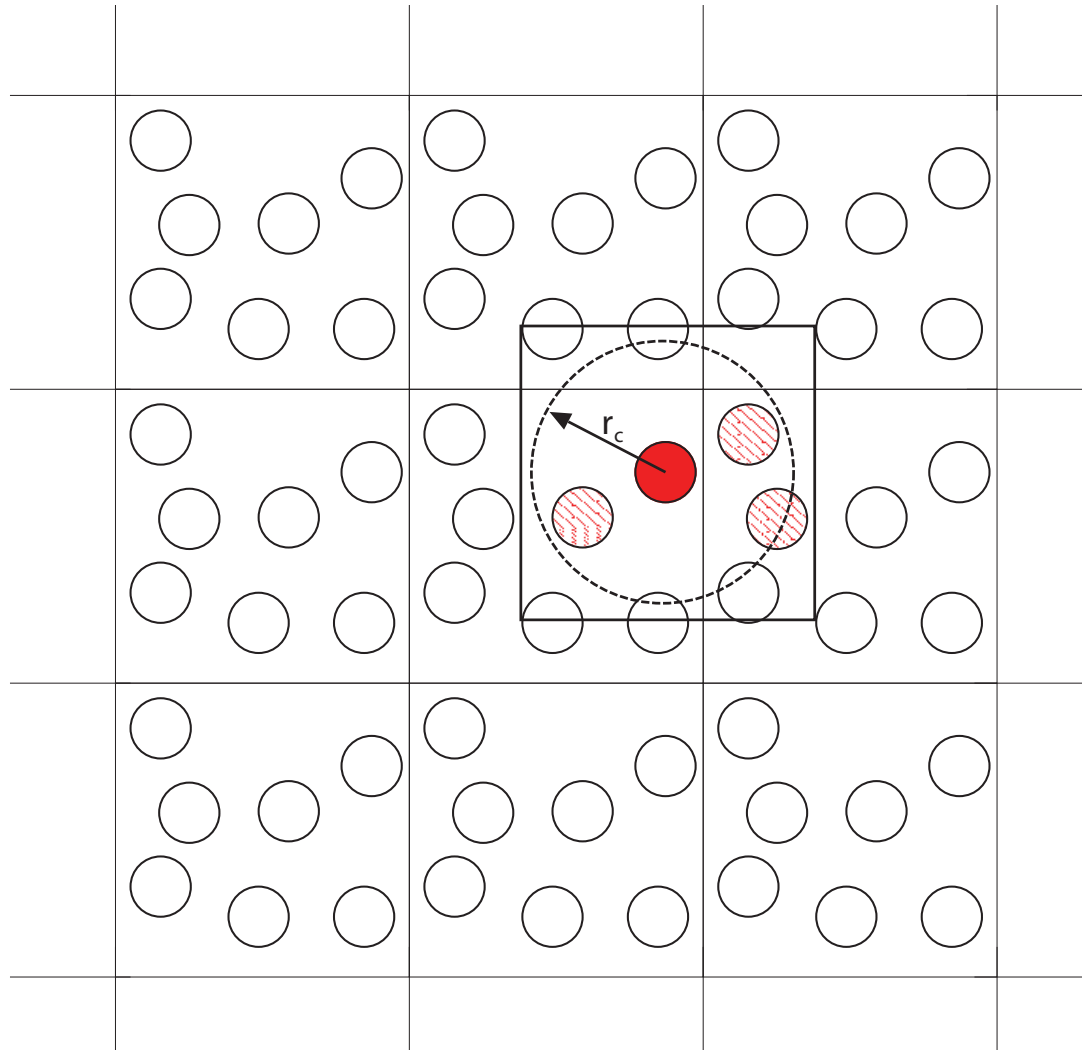
- ▶ Disadvantage: Discontinuity in the energy and forces

- ▶ Switching function ( $S(r_{ij})$  smoothly from 1 to 0)

$$V'_{ij}(r_{ij}) = V_{ij}(r_{ij}) \quad r_{ij} < r_l$$
$$V'_{ij}(r_{ij}) = S(r_{ij})V_{ij}(r_{ij}) \quad r_l \leq r_{ij} \leq r_c$$
$$V'_{ij}(r_{ij}) = 0 \quad r_{ij} > r_c$$

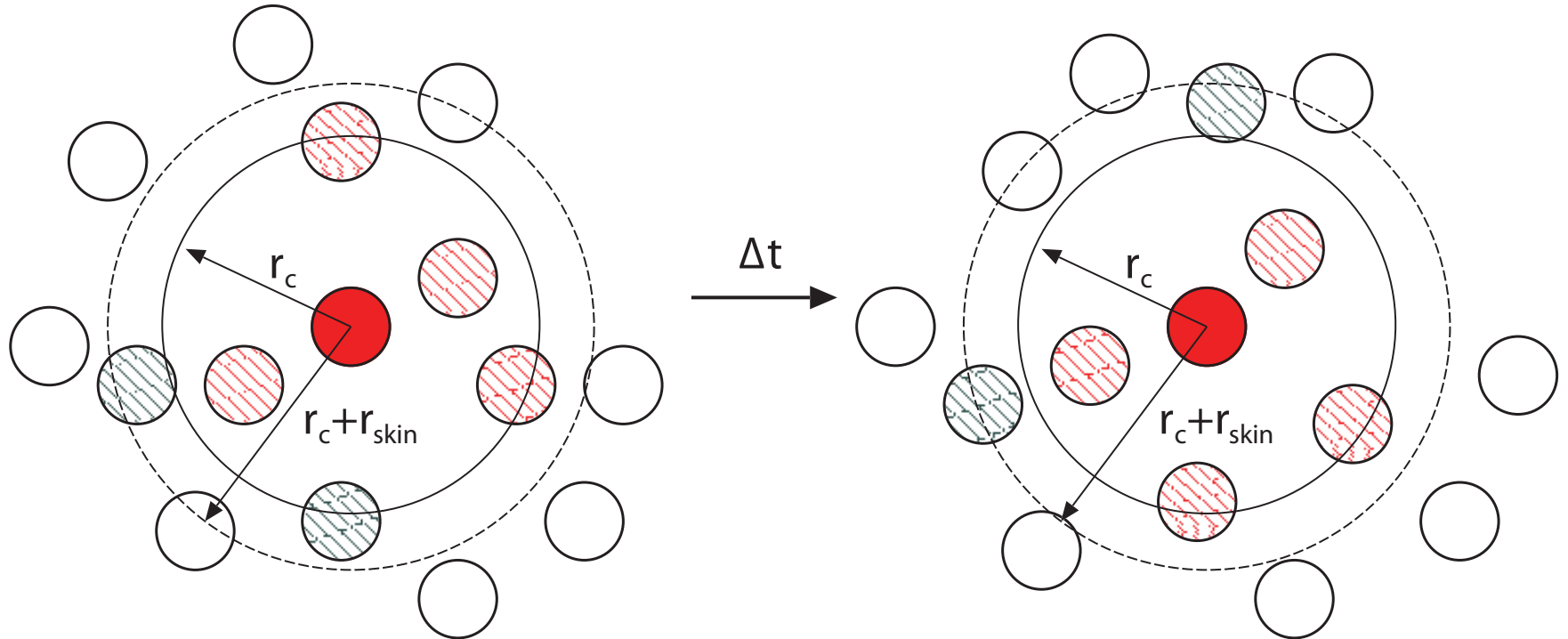
# Minimum Image and Spherical Cutoff

- ▶ Very commonly minimum images and spherical cutoffs are used together



# Searching for Neighbors

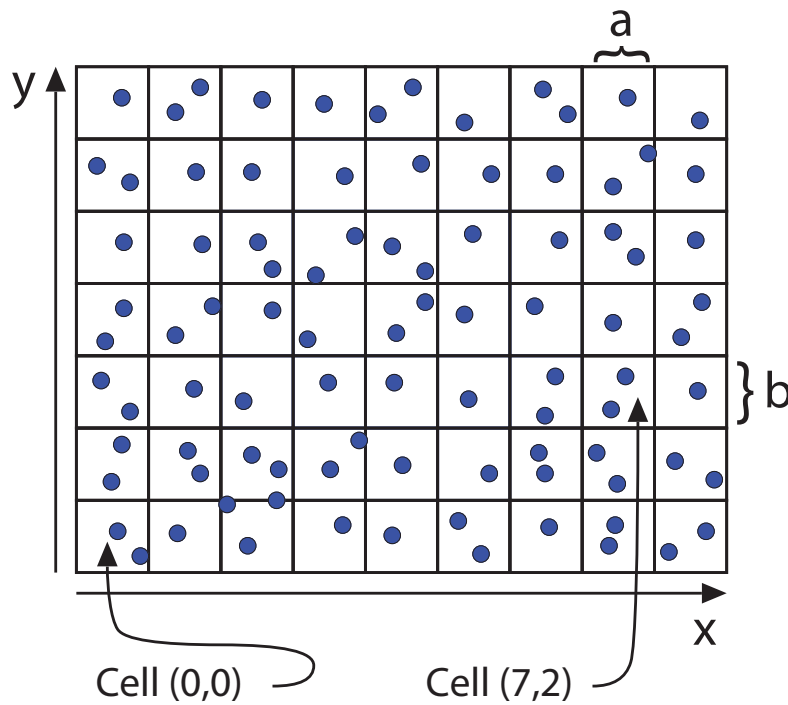
- ▶ The search loop is a double loop over all particles ( $n$ ). Execution time for the loop is proportional to  $n^2$
- ▶ The Verlet neighbor list: Maintain list of neighbor pairs closer than the radius ( $r_c$ ) plus a buffer ( $r_{skin}$ )



- ▶ Update list from time to time (about once every 25 timesteps, but this can be automated)

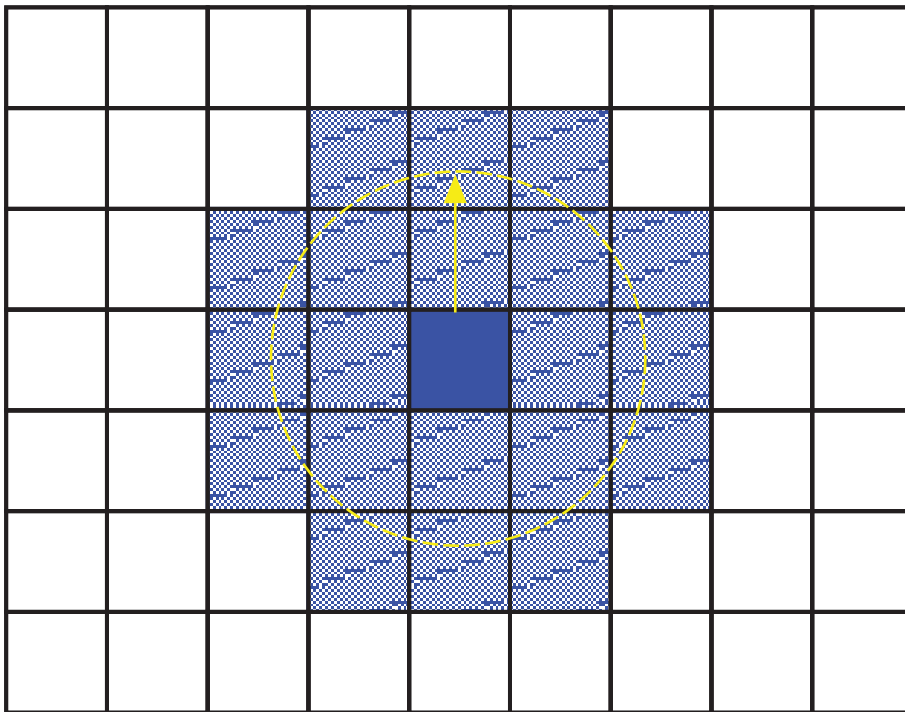
# Use of Linked Lists (Cell Method)

- ▶ For large systems the execution time will be dominated by the search loop (because building the Verlet neighbor lists are still  $O(N^2)$ )
- ▶ One solution is to divide the space into cells



- ▶ Finding which cell a particle belongs to is an  $O(N)$  problem. A particle with coordinate  $(x, y)$  belongs to cell  $(\text{floor}(x/a), \text{floor}(y/b))$
- ▶ All particles are sorted into so-called linked lists. There is one list for each cell.

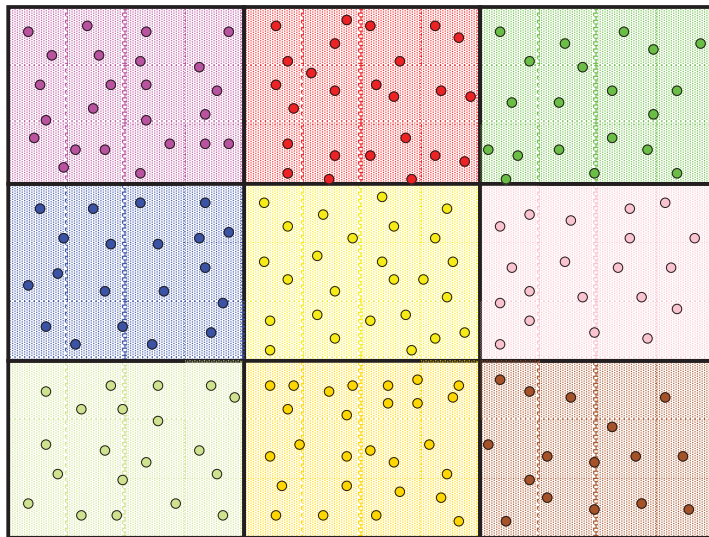
- ▶ When all particles are sorted into lists, the double loop can be done over cells instead of particles. We don't have to consider all cells. Only cells with particles closer to each other than the cutoff distance must be considered :



- ▶ For the particles inside the central cell only particles inside the central cell itself and the shadowed cells are considered if the cutoff distance is equal to, for instance, 1.2 cell lengths

# Domain Decomposition

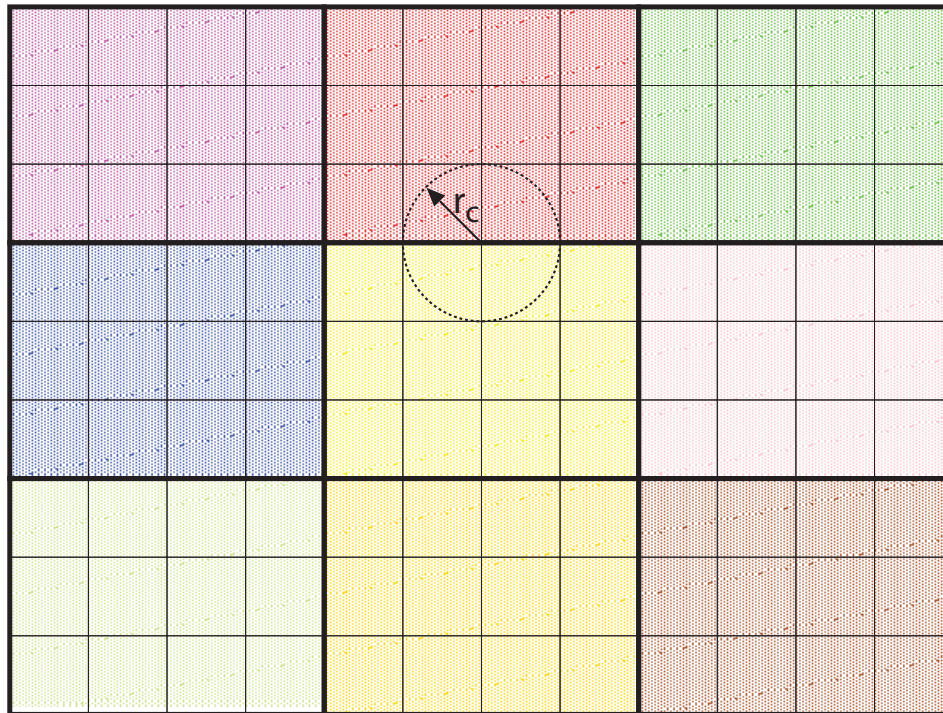
- ▶ For very large systems the memory requirement on each node becomes large, when using a replicated data algorithm
- ▶ Also the program will become slow due to that forces from all particles must be sent to all processors each timestep

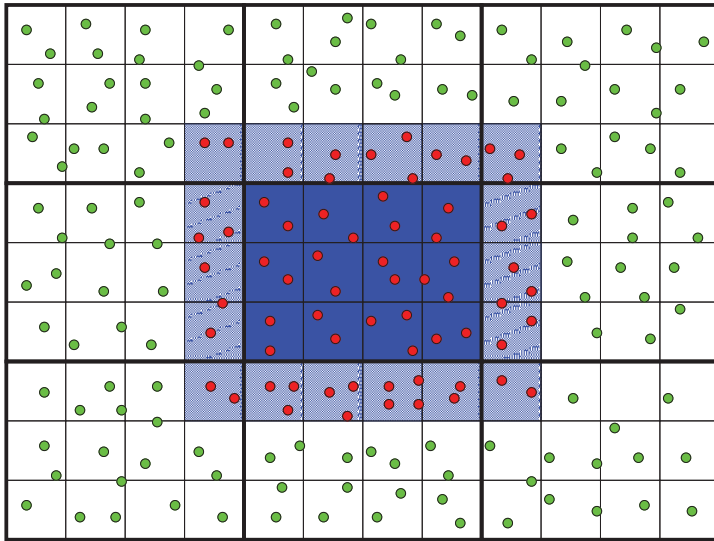


A 3x3 decomposition of a two dimensional system

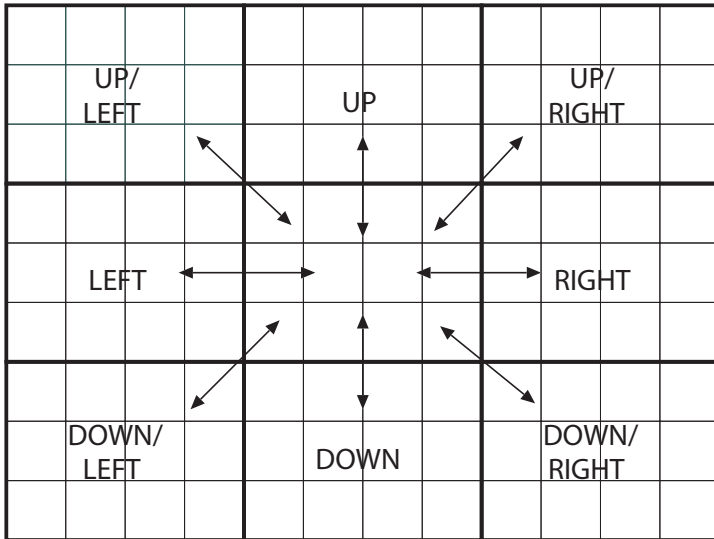
- ▶ One solution for very large systems is to use domain decomposition. Particles are assigned to processors depending on their position in space.
- ▶ When domain decomposition is used larger systems may be simulated simply by using more processors

- ▶ For large systems the interaction cutoff radius,  $r_c$ , is substantially smaller than the size of the system





- ▶ Particles closer to the “edge” of the domain than the cutoff radius, interact with particles on the node.



- ▶ 8 neighbours in 2D,  
and 26 in 3D

# Integrating the Equations of Motion

- ▶ Motion of all particles is coupled (many-body problem), This is impossible to solve analytically ...
- ▶ Solve using finite differences
- ▶ Positions, velocities ... approximated as a Taylor series:

$$r(t + \Delta t) = r(t) + \Delta t v(t) + \frac{1}{2} \Delta t^2 a(t) + \frac{1}{6} \Delta t^3 b(t) + \frac{1}{24} \Delta t^4 c(t) + \dots$$

$$v(t + \Delta t) = v(t) + \Delta t a(t) + \frac{1}{2} \Delta t^2 b(t) + \frac{1}{6} \Delta t^3 c(t) + \dots$$

$$a(t + \Delta t) = a(t) + \Delta t b(t) + \frac{1}{2} \Delta t^2 c(t) + \dots$$

$$b(t + \Delta t) = b(t) + \Delta t c(t) + \dots$$

- ▶ The simplest method truncate the series after the first-order term. This gives Euler's method :

$$\begin{aligned}r(t + \Delta t) &= r(t) + \Delta t v(t) \\v(t + \Delta t) &= v(t) + \Delta t a(t)\end{aligned}$$

- ▶ This is a first order method (and should not be used)
- ▶ The Verlet algorithm (1968, Loop Verlet) :

$$\begin{aligned}r(t + \Delta t) &= r(t) + \Delta t v(t) + \frac{1}{2} \Delta t^2 a(t) \\r(t - \Delta t) &= r(t) - \Delta t v(t) + \frac{1}{2} \Delta t^2 a(t)\end{aligned}$$

- ▶ Add the equations, and rearrange :

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + \Delta t^2 a(t)$$

- ▶ The Verlet algorithm is a second order algorithm
- ▶ Drawbacks of the Verlet algorithm :
  - ▶ Velocities do not appear as they have been eliminated by the addition. They must be estimated, for instance by taking another integration step :

$$v(t) = \frac{r(t + \Delta t) - r(t - \Delta t)}{2 \Delta t}$$

Velocities are needed, for instance, when computing the kinetic energy of the system

- ▶ It is not a self-starting algorithm
- ▶ A small term ( $\Delta t^2 a(t)$ ) is added to the difference of two large terms ( $2 r(t) - r(t - \Delta t)$ )

▶ There are several variations of the Verlet method :

▶ The Leap-Frog Verlet method :

$$r(t + \Delta t) = r(t) + \Delta t v(t + \frac{1}{2}\Delta t)$$
$$v(t + \frac{1}{2}\Delta t) = v(t - \frac{1}{2}\Delta t) + \Delta t a(t)$$

▶ Disadvantage: Velocities and positions not available at the same time

▶ The Velocity Verlet method :

$$r(t + \Delta t) = r(t) + \Delta t v(t) + \frac{1}{2}\Delta t^2 a(t)$$
$$v(t + \Delta t) = v(t) + \frac{\Delta t}{2} [a(t) + a(t + \Delta t)]$$

▶ Solved in two steps with a force evaluation in between

▶ Yields the positions, velocities and accelerations at the same time

- ▶ Apart from numerical differences, the three Verlet methods result in the same trajectories
- ▶ The Gear predictor-corrector integrator :
  - ▶ First new positions, velocities and accelerations are predicted according to the Taylor expansions :

$$r^p(t + \Delta t) = r(t) + \Delta t v(t) + \frac{1}{2} \Delta t^2 a(t) + \frac{1}{6} \Delta t^3 b(t) + \frac{1}{24} \Delta t^4 c(t)$$

$$v^p(t + \Delta t) = v(t) + \Delta t a(t) + \frac{1}{2} \Delta t^2 b(t) + \frac{1}{6} \Delta t^3 c(t)$$

$$a^p(t + \Delta t) = a(t) + \Delta t b(t) + \frac{1}{2} \Delta t^2 c(t)$$

$$b^p(t + \Delta t) = b(t) + \Delta t c(t)$$

- ▶ Then the forces are evaluated to give the true accelerations  $a(t + \Delta t)$

- ▶ The Gear predictor-corrector integrator (cont) :
  - ▶ The difference between the predicted and calculated accelerations is computed :

$$\Delta a(t + \Delta t) = a(t + \Delta t) - a^p(t + \Delta t)$$

- ▶ The correction step :

$$r^c(t + \Delta t) = r^p(t + \Delta t) + c_0 \Delta a(t + \Delta t)$$

$$v^c(t + \Delta t) = v^p(t + \Delta t) + c_1 \Delta a(t + \Delta t)$$

$$a^c(t + \Delta t) = a^p(t + \Delta t) + c_2 \Delta a(t + \Delta t)$$

$$b^c(t + \Delta t) = b^p(t + \Delta t) + c_3 \Delta a(t + \Delta t)$$

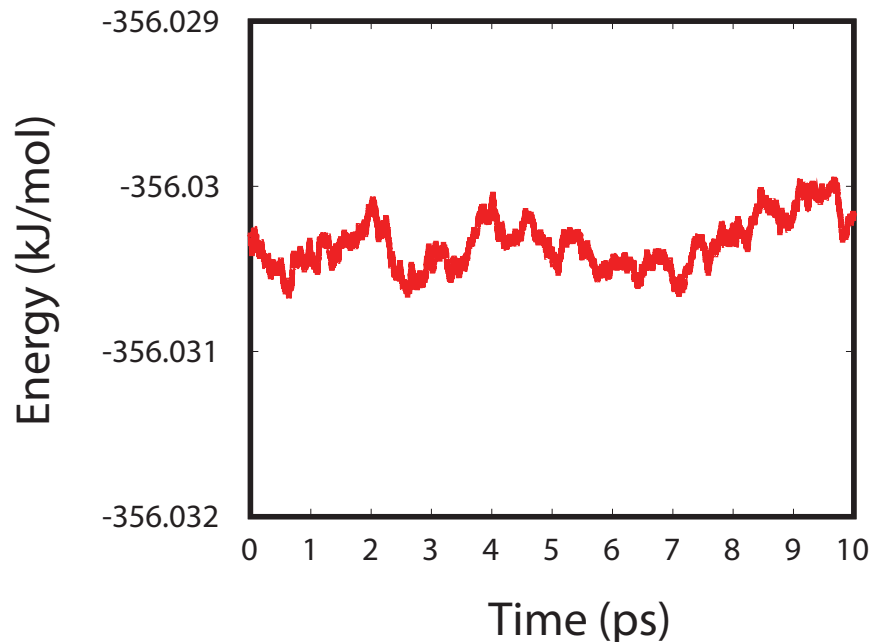
$$c^c(t + \Delta t) = c^p(t + \Delta t) + c_4 \Delta a(t + \Delta t)$$

- ▶ The values of the coefficients  $c_0, c_1, \dots$  depend on the order of the method (  $c_0 = \frac{19}{120}, c_1 = \frac{3}{4}, c_2 = 1, c_3 = \frac{1}{2}, c_4 = \frac{1}{12}$  )

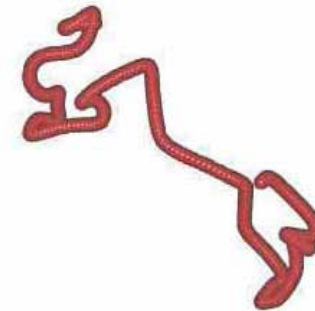
## ▶ Choosing the Timestep

- ▶ A balance is needed in choosing the timestep. A too short timestep lead to a very expensive solution of the equations of motion, and leads to a limited coverage of phase space. A much too small timestep might lead to round-off errors.
- ▶ Too large of a timestep leads to instabilities

## ▶ Example: Simulations of the 2D Lennard-Jones Argon System



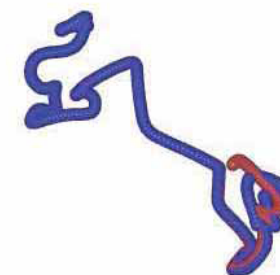
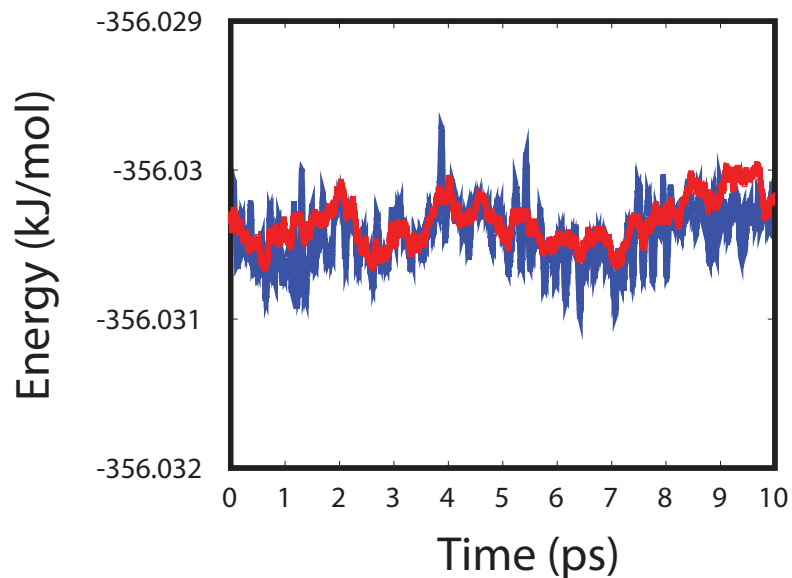
Total energy as a function of time



“Exact” trajectory of one argon atom

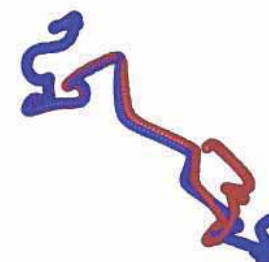
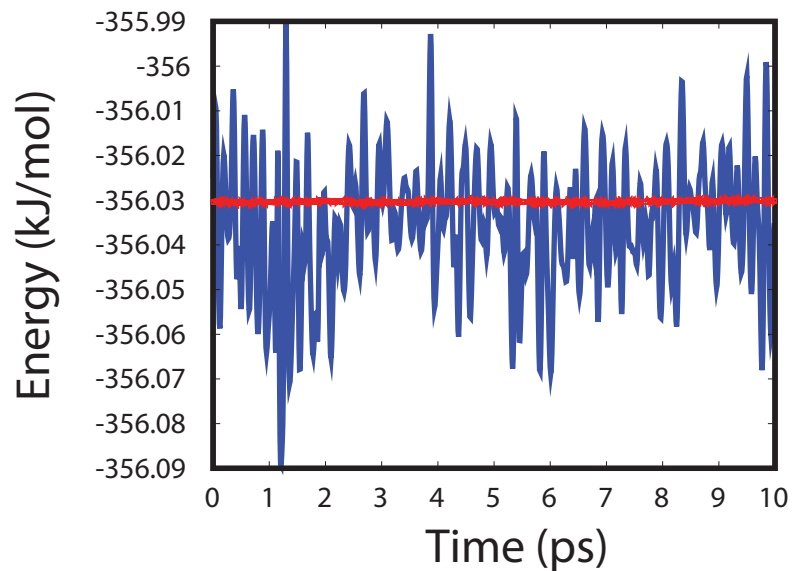
▶ Velocity Verlet Integrator

$\Delta t = 1$  fs (blue)



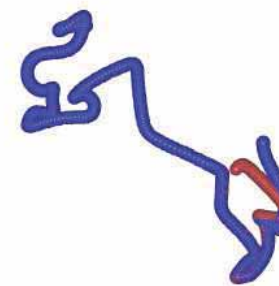
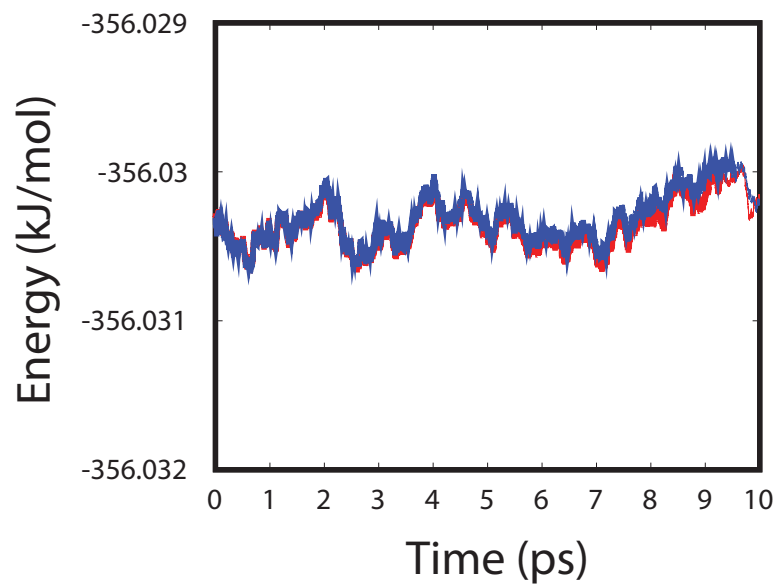
▶ Velocity Verlet Integrator

$\Delta t = 10$  fs (blue)



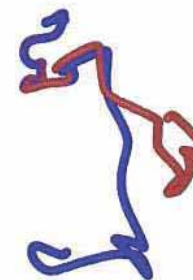
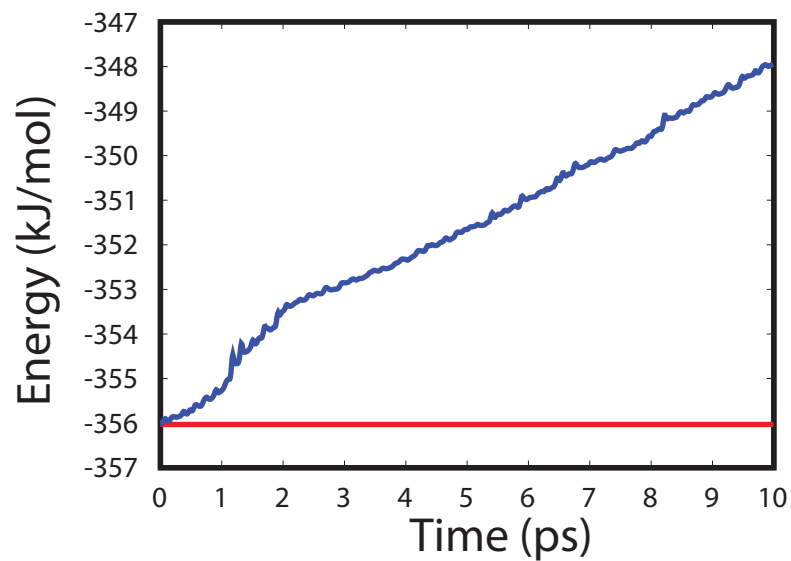
► Gear Predictor-Corrector Method

$\Delta t = 1$  fs (blue)

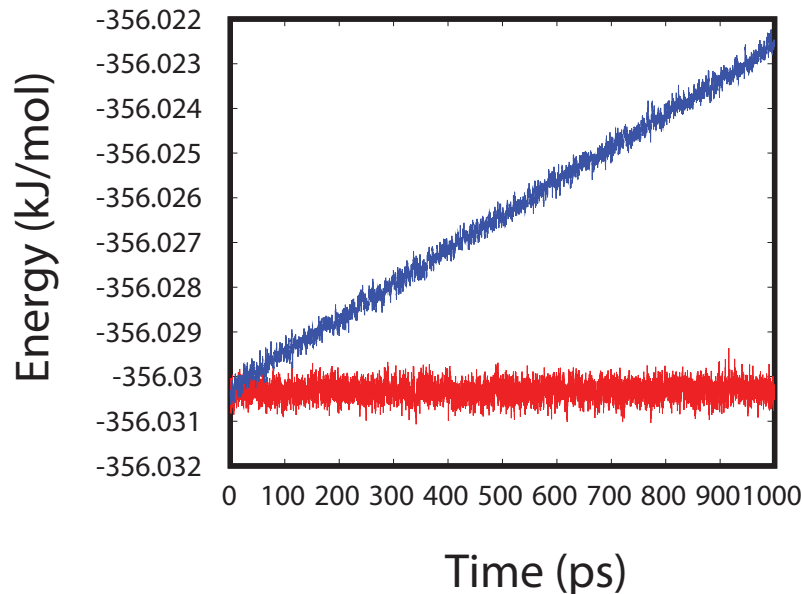


► Gear Predictor-Corrector Method

$\Delta t = 10$  fs (blue)



- ▶ Comparison of Velocity Verlet (red) and the Gear Predictor-Corrector (blue) at  $\Delta t = 1$  fs :



- ▶ The Gear method has short time fluctuations, but it has a (substantial) long-time drift
  - ▶ The Verlet method uses a central difference and is thus time-reversible
- 
- ▶ The trajectories will eventually always diverge from the “true” trajectory
  - ▶ The energy (Hamiltonian) of the system should be conserved
  - ▶ Use a shorter timestep for :
    - ▶ Lighter particles
    - ▶ Higher temperatures
    - ▶ Stiffer potentials