Basic Monte Carlo (chapter 3)

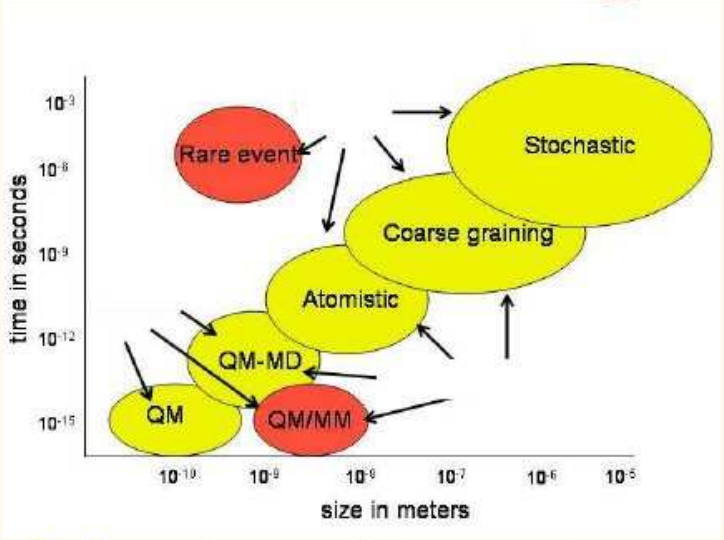
Algorithm

Detailed Balance

Other points

CONTEXT

Perspective: Time and Length Scales



- Correlated Methods: $\sim$ 1 ps, $\sim$ 0.1 nm
- DFT: $\sim$ 10 ps, $\sim$ nm
- Hybrid DFT/empirical: $\sim$ 10 ps, $\sim$ 10 nm
- Atomistic semi-empirical: $\sim$ 100 ps, $\sim$ 10 nm
- Atomistic empirical: $\sim$ 10 ns, $\sim$ 100 nm
- Coarse grained particle models: $\sim \mu$s, $\sim \mu$m
- Continueous models: $\sim$ s, $\sim$ m

CONTEXT

Goal

Understand System on a Atomistic/Particle Level

Properties

- Equation of State
- Local Structure
- Electronic Properties
- Phase Equilibria
- Dynamics of Transitions
- ....

Conditions

- Temperature
- Pressure
- pH
- Electric Fiel
- Compositions
- ...

CONTEXT

FORCES

FORCE-FIELD APPROACH

EMPERICAL AND/OR FITTED INTER-ATOMIC FORCES

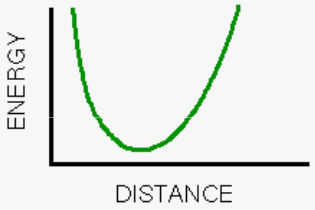
INTRA-MOLECULAR:

VIBRATION → HARMONIC, OR BETTER

BEND

TORSION

....



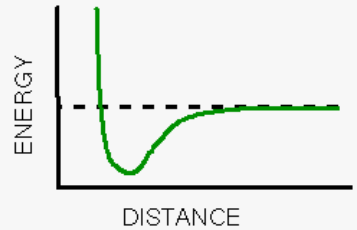
INTER-MOLECULAR:

STERIC REPULSION
DISPERSION
} → LENNARD-JONES, OR BETTER

HYDROGEN BONDS

ELECTROSTATIC

...

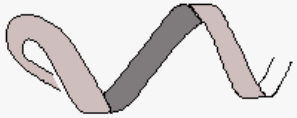


CONTEXT

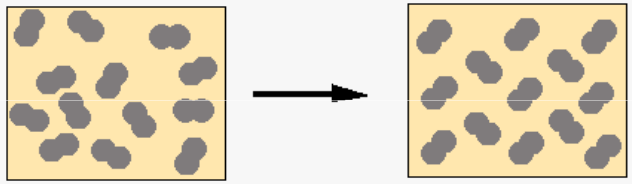
FORCE FIELDS

FOR MANY SYSTEMS: ACCURATE AND EFFICIENT

STRUCTURE



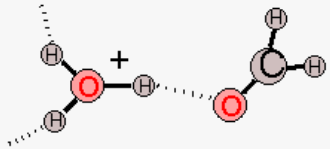
PHASE BEHAVIOR



FOR CHEMICAL REACTIVE SYSTEMS: DIFFICULT OR IMPOSSIBLE

INTRA-MOLECULAR INTER-MOLECULAR

CHANGING ELECTRONIC STRUCTURE



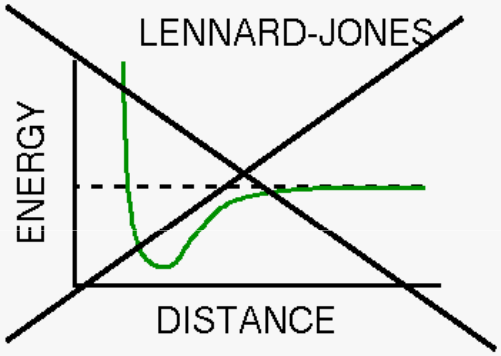
FORCE FIELDS NOT SUFFICIENTLY TRANSFERABLE

CONTEXT

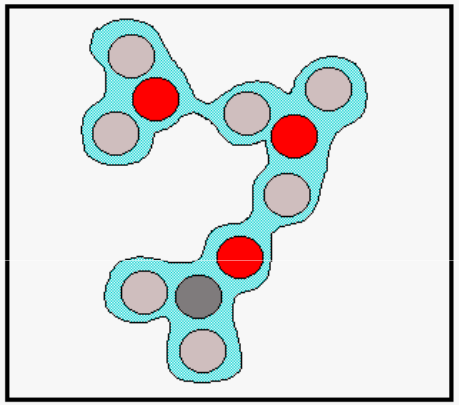
FORCES

FIRST-PRINCIPLES APPROACH

INTER-ATOMIC FORCES BY ELECTRONIC STRUCTURE CALCULATION



BUT



CHANGING CHEMISTRY IMPLICIT

DIFFICULT (MANY ELECTRONS)

COMPUTATIONALLY EXPENSIVE

NOT ADDRESSED IN THIS SCHOOL

APPROACH

Monte Carlo and Molecular Dynamics

Generate a representative set of configurations or trajectory of the N particle system

Monte Carlo

- Sample Boltzmann distribution by stochastic generation of configurations
- Efficient but limited dynamical properties

Molecular Dynamics

- Generate trajectory by solving equation of motion numerically
- Provides explicit time-dependence
- Proper averages require ergodicity

Statistical Thermodynamics

Partition function

$$Q_{NVT} = \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}^N \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

Ensemble average

$$\left\langle A \right\rangle_{NVT} = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}^N A\left(\mathrm{r}^N\right) \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

Probability to find a particular configuration

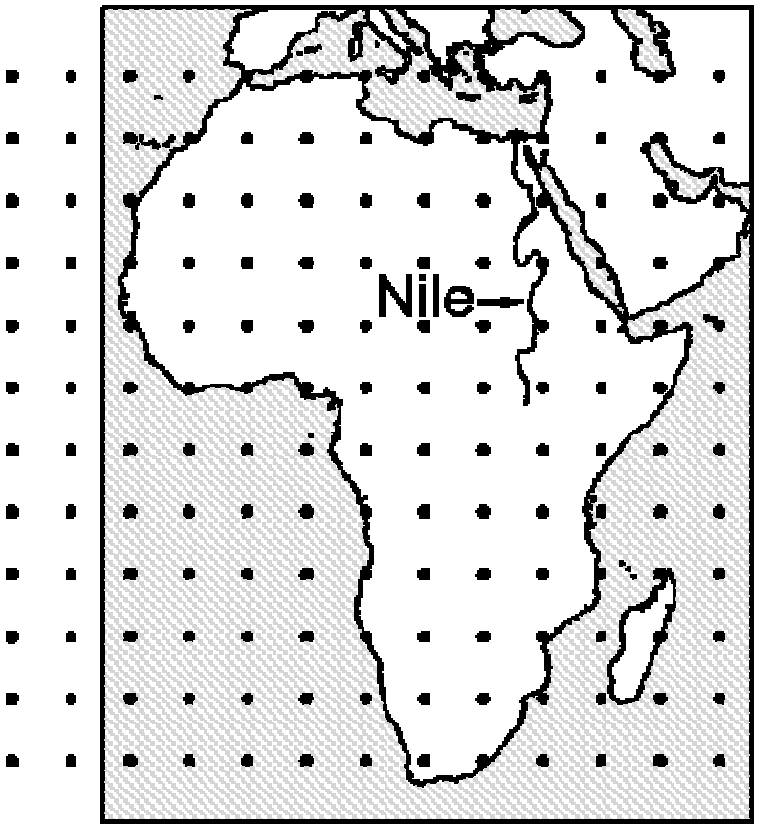
$$\mathrm{N}\left(\mathrm{r}^N\right) = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}'^N \delta\left(\mathrm{r}'^N - \mathrm{r}^N\right) \exp\left[-\beta U\left(\mathrm{r}'^N\right)\right] \propto \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

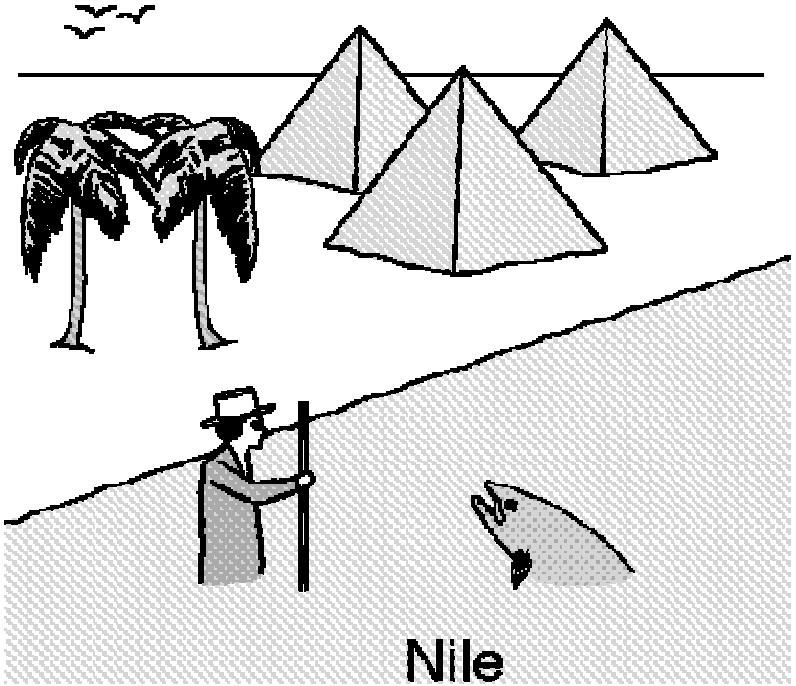
Free energy

$$\beta F = -\ln\left(Q_{NVT}\right)$$

Monte Carlo simulation

Measure the Average Depth of the Nile





Ensemble average

$$\langle A\rangle_{NVT} = \frac{1}{Q_{NVT}}\frac{1}{\Lambda^{3N}N!}\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)\exp\left[-\beta U(\mathbf{r}^N)\right]$$

$$= \int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)P(\mathbf{r}^N) = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)P(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N P(\mathbf{r}^N)}$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)C\exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N C\exp\left[-\beta U(\mathbf{r}^N)\right]} = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)\exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

$$P(\mathbf{r}^N) = \frac{\exp\left[-\beta U(\mathbf{r}^N)\right]}{Q_{NVT}\Lambda^{3N}N!}$$

Generate configuration using MC:

$$\{\mathbf{r}_1^N, \mathbf{r}_2^N, \mathbf{r}_3^N, \mathbf{r}_4^N \quad , \mathbf{r}_M^N\}$$

$$\bar{A} = \frac{1}{M}\sum_{i=1}^{M} A(\mathbf{r}_i^N) = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)P^{MC}(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N P^{MC}(\mathbf{r}^N)}$$

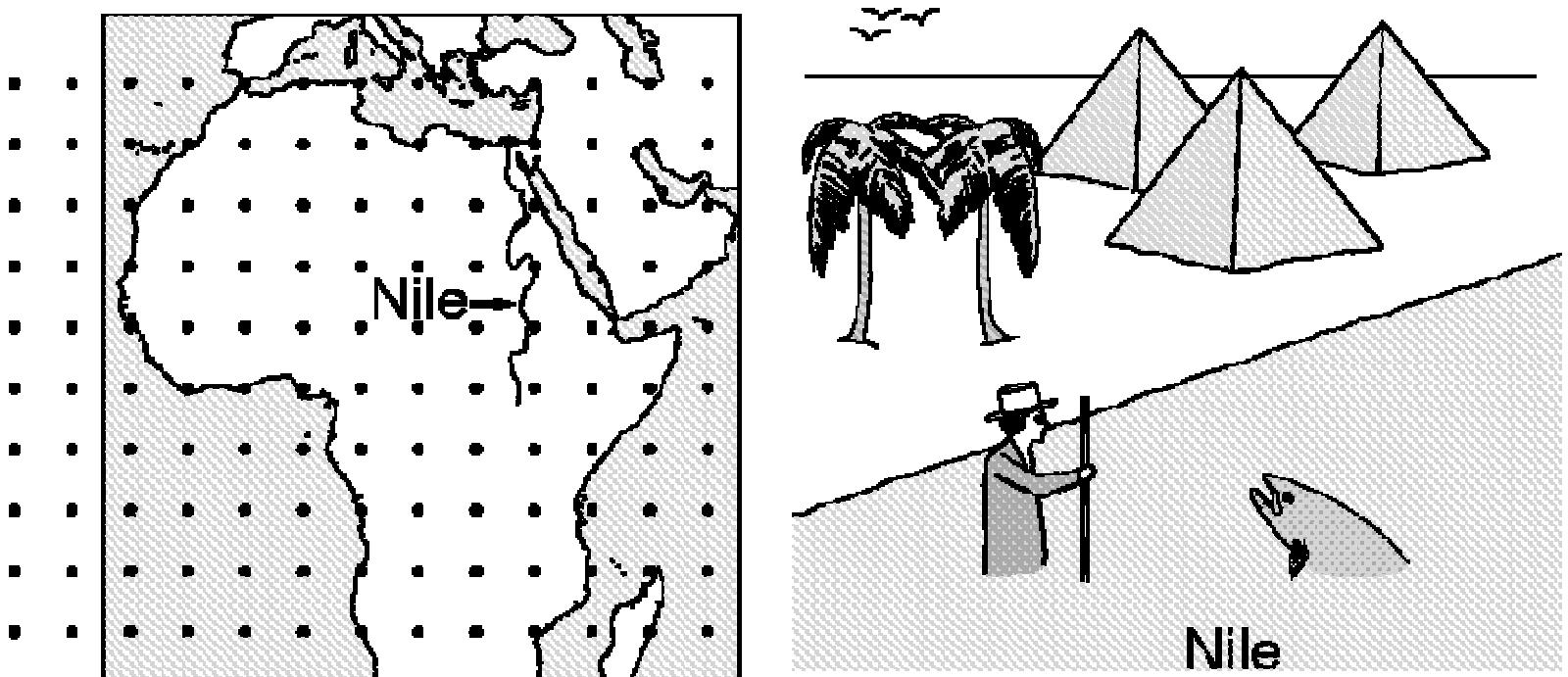
with

$$P^{MC}(\mathbf{r}^N) = C^{MC}\exp\left[-\beta U(\mathbf{r}^N)\right]$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)C^{MC}\exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N C^{MC}\exp\left[-\beta U(\mathbf{r}^N)\right]}$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N)\exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

Monte Carlo simulation



Brute-force Monte Carlo ("Random sampling")

$$\int_V dx_1 \cdots dx_N f(x) \approx V^N \sum_{i-1}^{M} f(r_i)$$

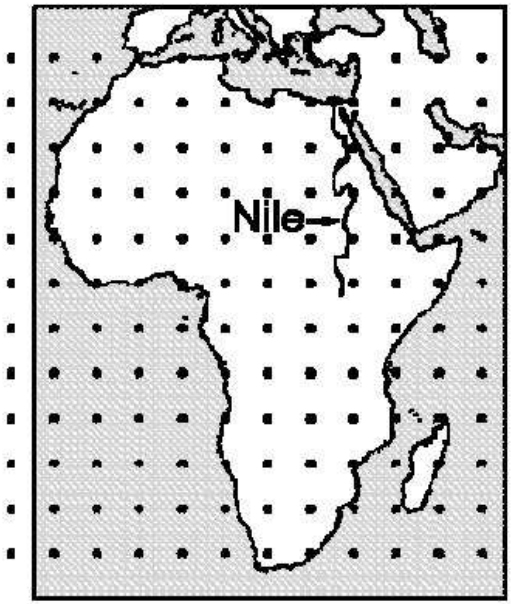
(M random points r_i in (hyper)volume V_N)

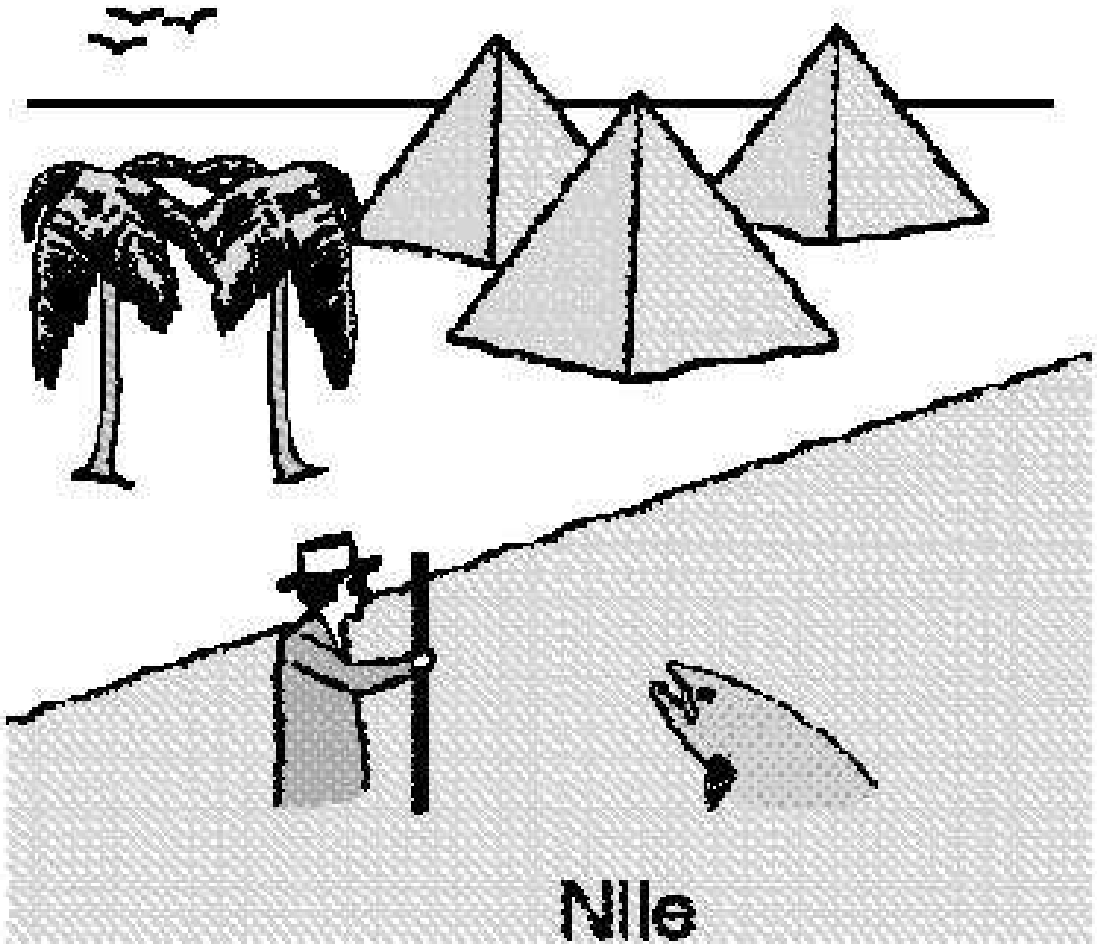
This Integral is a Problem

Can **not** be computed <u>analytically</u>,
except for a few models

<u>Numerical</u> evaluation inaccurate or "endless": ⟷
100 particles, 3 dimensions,
10 points per degree of freedom:
10^{300} points (poor estimate)

© D. Frenkel





BETTER
STRATEGY:

IMPORTANCE

SAMPLING

Algorithm 1 (Basic Metropolis Algorithm) (Frenkel & Smit: p. 33)

```
 PROGRAM mc                                  basic Metropolis algorithm

 do icycl=1,ncycl                            perform ncycl MC cycles
   call mcmove                               displace a particle
   if (mod(icycl,nsamp).eq.0)
+    call sample                            sample averages
 enddo
 end
```

Comments to this algorithm:

1. *Subroutine* `mcmove` *attempts to displace a randomly selected particle (see Algorithm 2).*
2. *Subroutine* `sample` *samples quantities every* `nsamp`*th cycle.*

Algorithm 2 (Attempt to Displace a Particle) (Frenkel & Smit: p. 33)

```
      SUBROUTINE mcmove                attempts to displace a particle

      o=int(ranf()*npart)+1            select a particle at random
      call ener(x(o),eno)              energy old configuration
      xn=x(o)+(ranf()-0.5)*delx        give particle random displacement
      call ener(xn,enn)                energy new configuration
      if (ranf().lt.exp(-beta          acceptance rule (3.2.1)
     +    *(enn-eno)) x(o)=xn          accepted: replace x(o) by xn
      return
      end
```

Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* $[0, 1]$.

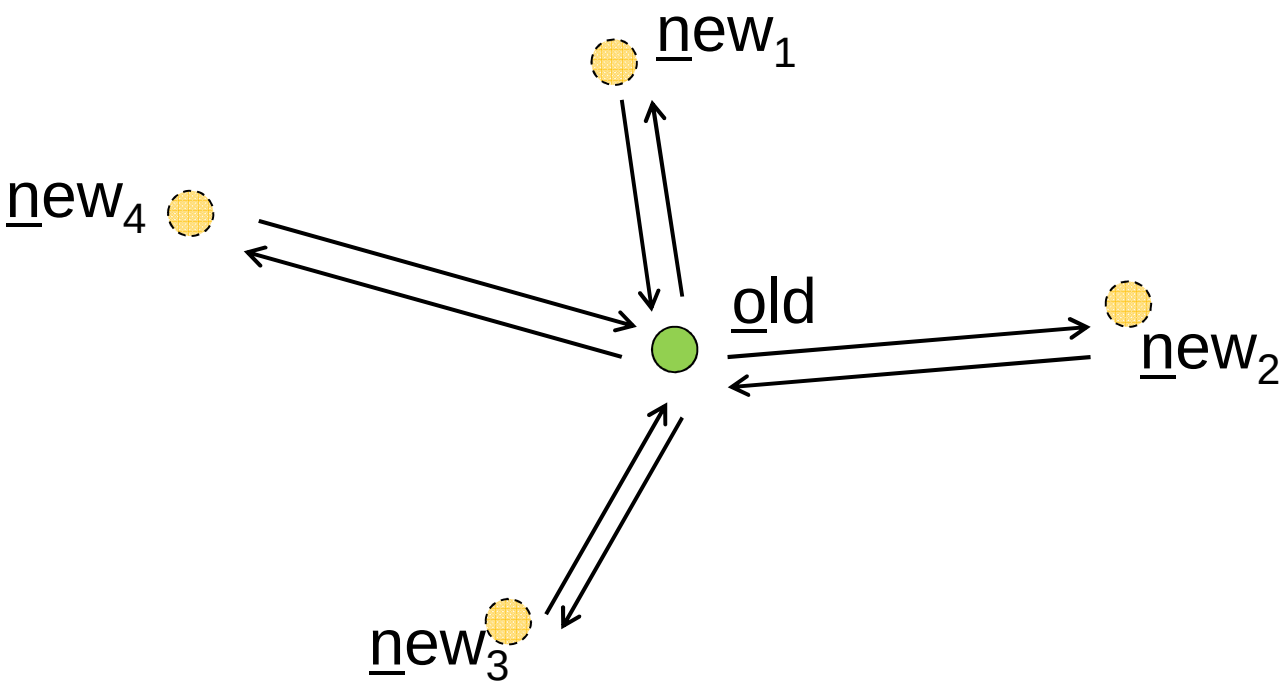
Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Central Requirement

Whatever the rule is for moving a particle
from one point to another,
it should not destroy the equilibrium distribution

Hence, in equilibrium we must have:



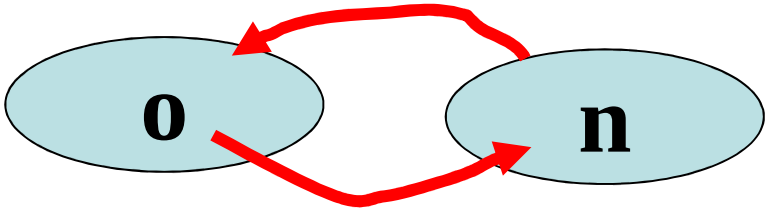
$\mathcal{N}(n)$ Probability Distribution of States

$\pi(o \to n)$ Transition Probability going from *o* to *n*

Equilibrium should be Conserved

$$\mathcal{N}(o)\sum_n \pi(o \to n) = \sum_n \mathcal{N}(n)\pi(n \to o)$$

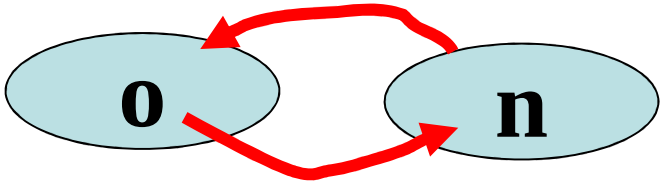
Stronger Condition



Detailed Balance

$$\mathcal{N}(o)\pi(o \to n) = \mathcal{N}(n)\pi(n \to o).$$

For every pair {n,o}



Construct Transition Probability

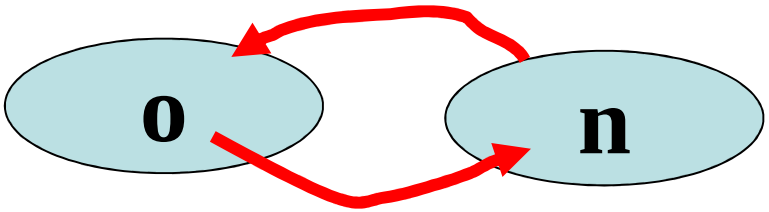
$\alpha(o \to n)$ Probability to attempt transition

$\mathrm{acc}(o \to n)$ Probability to accept transition

$$\pi(o \to n) = \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

Detailed Balance Implies

$$\mathcal{N}(o)\alpha(o \to n) \times \mathrm{acc}(o \to n)$$
$$=$$
$$\mathcal{N}(n)\alpha(n \to o) \times \mathrm{acc}(n \to o)$$

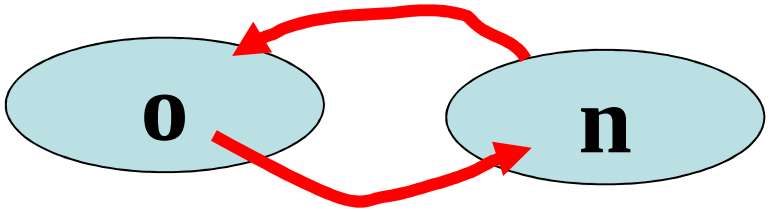


Often Transition Matrix Chosen Symmetric:

$$\alpha(o \to n) = \alpha(n \to o)$$

This Implies (with Detailed Balance)

$$\mathcal{N}(o) \times \mathrm{acc}(o \to n) \quad = \quad \mathcal{N}(n) \times \mathrm{acc}(n \to o)$$



NVT-ensemble

$$N(n) \propto \exp\left[-\beta U(n)\right]$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} = \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}$$

Metropolis, Rosenbluth,Rosenbluth,

Teller and Teller[#]choice:

[#] J. Chem. Phys. 21, 1087 (1953)

$$\mathrm{acc}(o \to n) = \begin{cases} \dfrac{\mathcal{N}(n)}{\mathcal{N}(o)} & \text{if} \quad \mathcal{N}(n) < \mathcal{N}(o) \\ 1 & \text{if} \quad \mathcal{N}(n) > \mathcal{N}(o) \end{cases}$$

$$= \min\left(1, \exp\{-\beta[\mathcal{U}(\mathbf{r}'^N) - \mathcal{U}(\mathbf{r}^N)]\}\right)$$

Importance Sampling Random Walk

A move starting from one point consist of generating a **trial move** and **accept or reject** such a move.

$$\mathrm{acc}(o \to n) = \min\left(1, e^{-\beta[U(n)-U(o)]}\right)$$

- try to change energy state
- compute $\Delta E = E_{new} - E_{old}$
- accept new state if ran < exp(ΔE/kT)
- reject otherwise
- sample the state of the system
- repeat

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Algorithm 2 (Attempt to Displace a Particle)

```
      SUBROUTINE mcmove              attempts to displace a particle

      o=int(ranf()*npart)+1          select a particle at random
      call ener(x(o),eno)            energy old configuration
      xn=x(o)+(ranf()-0.5)*delx      give particle random displacement
      call ener(xn,enn)              energy new configuration
      if (ranf().lt.exp(-beta        acceptance rule (3.2.1)
     +    *(enn-eno)) x(o)=xn        accepted: replace x(o) by xn
      return
      end
```

Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* [0, 1].

Recall:

Often Transition Matrix Chosen Symmetric:

$$\alpha(o \to n) = \alpha(n \to o)$$

Forward and Backward Move Equally Probably -> Random Selection

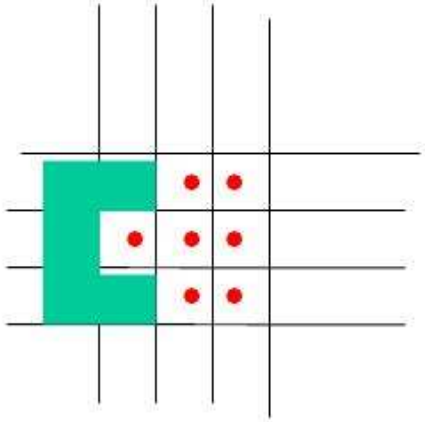
Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

History

Kirkwood's objection:

"If a trial move has been rejected, one should not count the original state AGAIN…"



Counter-example:

Ideal gas on a lattice.

Mathematical

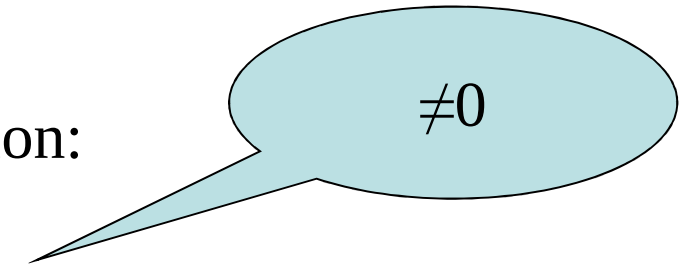
Transition probability:

$$\pi(o \to n) = \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$\sum_n \pi(o \to n) = 1$$

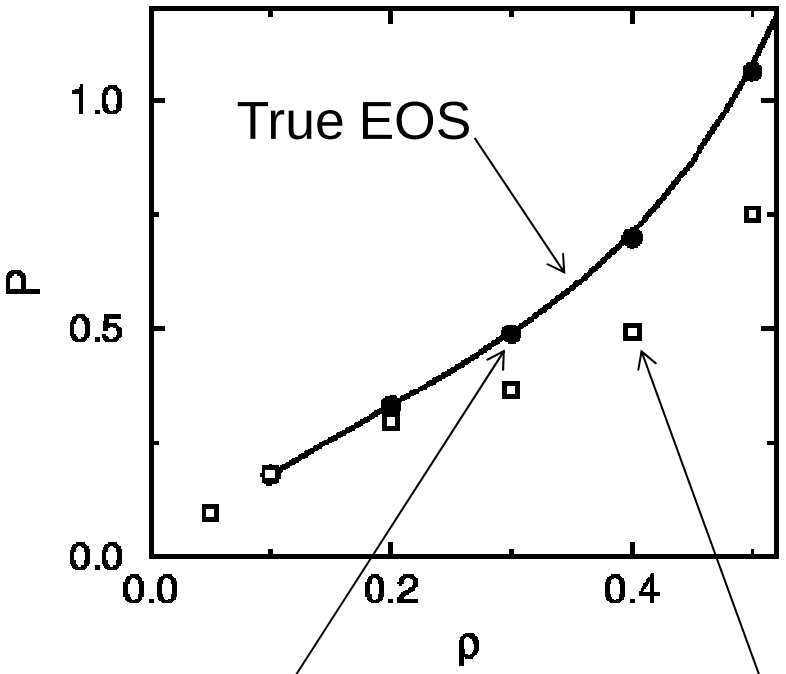
Probability to accept the old configuration:

$$\pi(o \to o) = 1 - \sum_{n \neq o} \pi(o \to n)$$



Keeping old configuration?

Lennard Jones Equation of State



MC: neglecting old configurations of rejected moves

MC: including old configurations of rejected moves

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Algorithm 2 (Attempt to Displace a Particle) (Frenkel & Smit: p. 33)

```
      SUBROUTINE mcmove                attempts to displace a particle

      o=int(ranf()*npart)+1            select a particle at random
      call ener(x(o),eno)              energy old configuration
      xn=x(o)+(ranf()-0.5)*delx        give particle random displacement
      call ener(xn,enn)                energy new configuration
      if (ranf().lt.exp(-beta          acceptance rule (3.2.1)
     +    *(enn-eno)) x(o)=xn          accepted: replace x(o) by xn
      return
      end
```

Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* $[0, 1]$.

Simulation in Practice

A Sampling Scheme

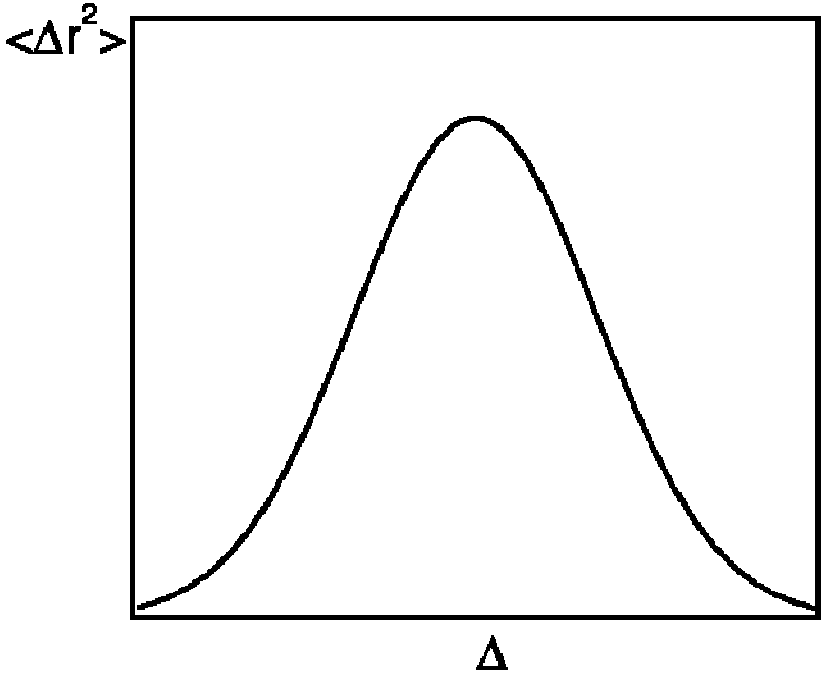
can be Considered Optimal if

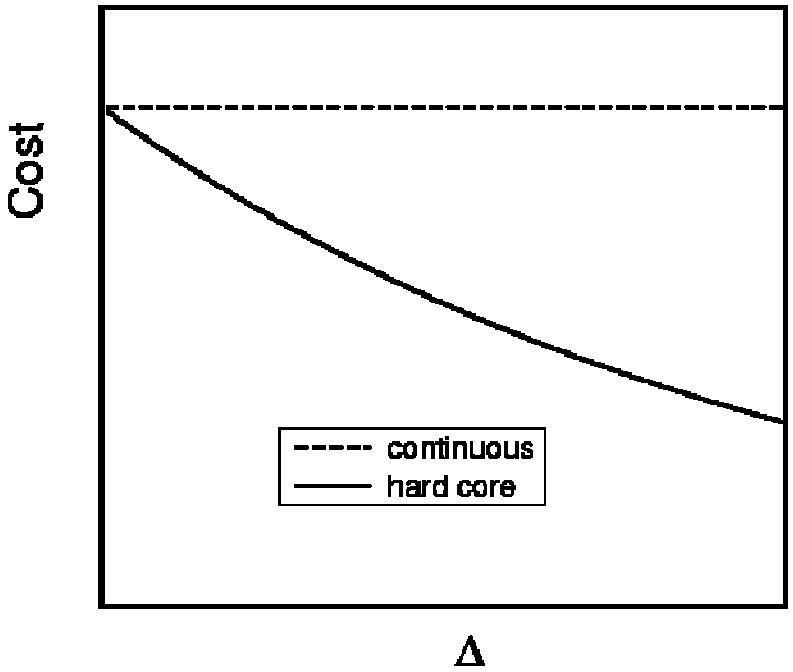
it Yields the Lowest Statistical Error

in the Quantity to be Computed

for a given Expenditure of Computing Budget

Not too small, not too big!





delx

Maximum attempted displacement

Optimal delx for hard core systems larger than for contineous potential systems

Algorithm 5 (Calculation of the Forces) and energies

```
subroutine force(f,en)                  determine the force
en=0                                    and energy
do i=1,npart
   f(i)=0                               set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                        loop over all pairs
     xr=x(i)-x(j)
     xr=xr-box*nint(xr/box)             periodic boundary conditions
     r2=xr**2
     if (r2.lt.rc2) then                test cutoff
       r2i=1/r2
       r6i=r2i**3
       ff=48*r2i*r6i*(r6i-0.5)          Lennard-Jones potential
       f(i)=f(i)+ff*xr                  update force
       f(j)=f(j)-ff*xr
       en=en+4*r6i*(r6i-1)-ecut         update energy
     endif
  enddo
enddo
return
end
```

Comments to this algorithm:

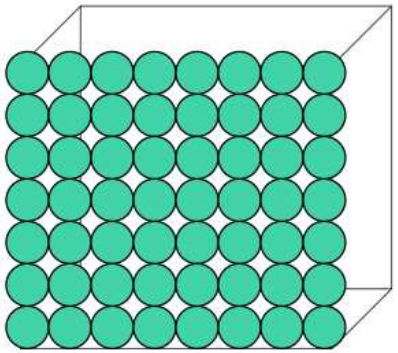
1. *For efficiency reasons the factors 4 and 48 are usually taken out of the force loop and taken into account at the end of the calculation for the energy.*
2. *The term* `ecut` *is the value of the potential at* $r = r_c$*; for the Lennard-Jones potential, we have*

$$\texttt{ecut} = 4\left(\frac{1}{r_c^{12}} - \frac{1}{r_c^{6}}\right).$$

Practical issues

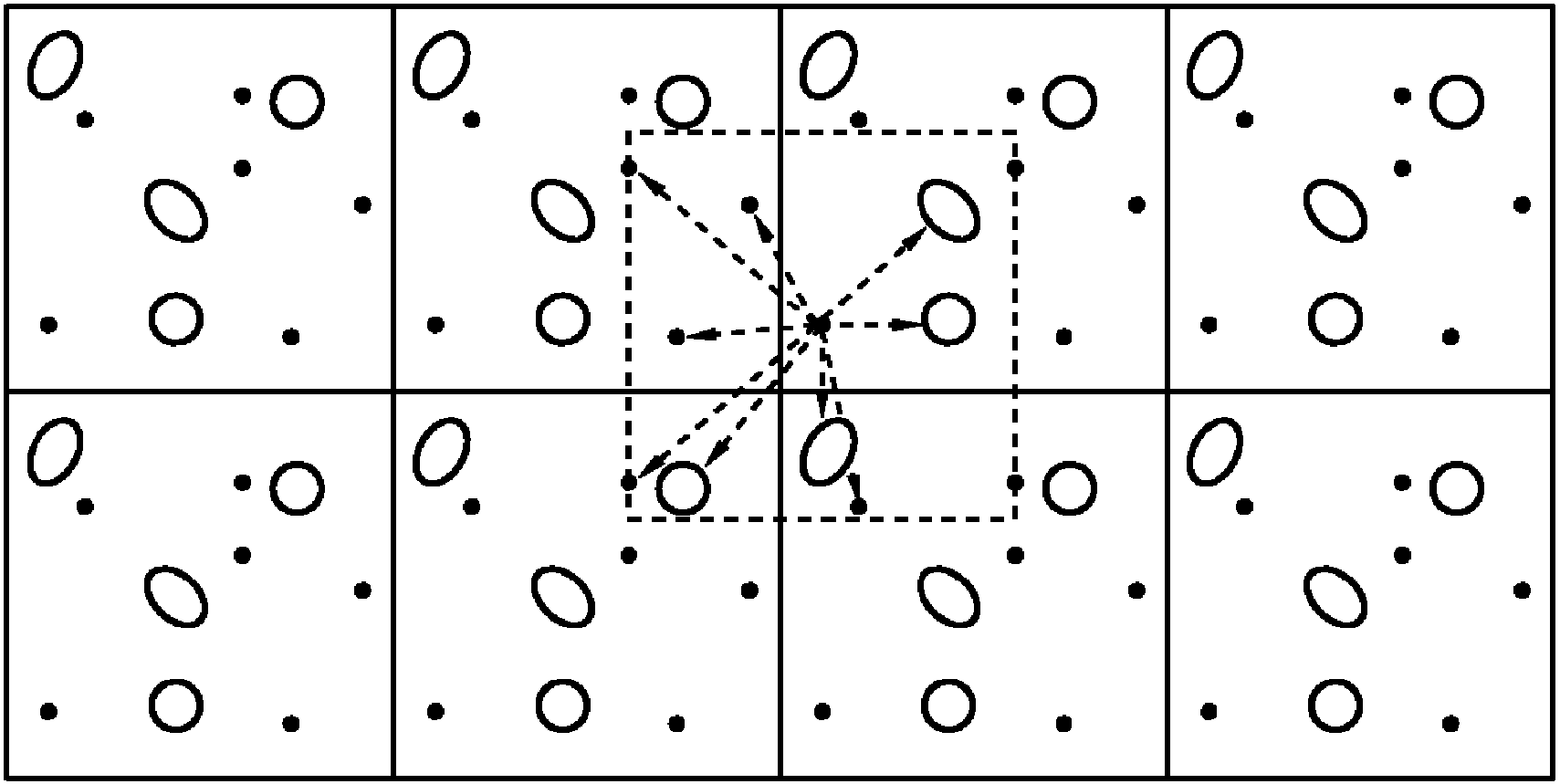
- Boundaries
- CPU saving methods
- Reduced units
- Long ranged forces

Boundary effects



- In small systems, boundary effects are always large.
- 1000 atoms in a simple cubic crystal – 488 boundary atoms.
- 1000000 atoms in a simple cubic crystal – still 6% boundary atoms.

Periodic boundary conditions



Algorithm 5 (Calculation of the energies)

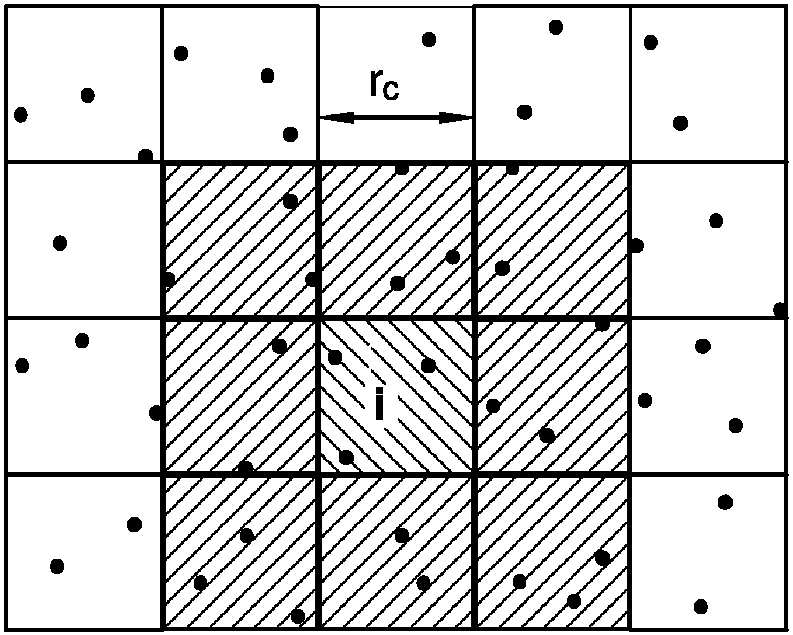
```
subroutine ener(x,en)                    determine the force
en=0                                     and energy
do i=1,npart
   f(i)=0                                set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                         loop over all pairs
    xr=x(i)-x(j)
    xr=xr-box*nint(xr/box)               periodic boundary conditions
    r2=xr**2
    if (r2.lt.rc2) then                  test cutoff
      r2i=1/r2
      r6i=r2i**3
      ff=48*r2i*r6i*(r6i-0.5)            Lennard-Jones potential
      f(i)=f(i)+ff*xr                    update force
      f(j)=f(j)-ff*xr
      en=en+4*r6i*(r6i-1)-ecut           update energy
    endif
  enddo
enddo
return
end
```

Energy evaluation costs!

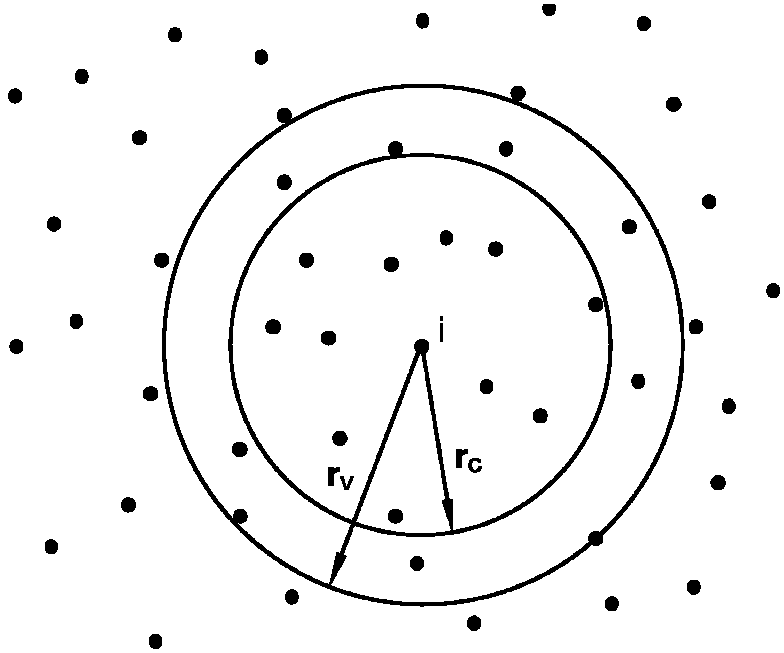
- The most time-consuming part of any simulation is the evaluation of all the interactions between the molecules.
- In general: $N(N-1)/2 = O(N^2)$
- But often, intermolecular forces have a short range:
- Therefore, we do not have to consider interactions with far- away atoms.

Saving CPU

- Cell list



Verlet List



Application: Lennard Jones potential

- The Lennard-Jones potential

$$u^{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

- The truncated Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) & r \le r_c \\ 0 & r > r_c \end{cases}$$

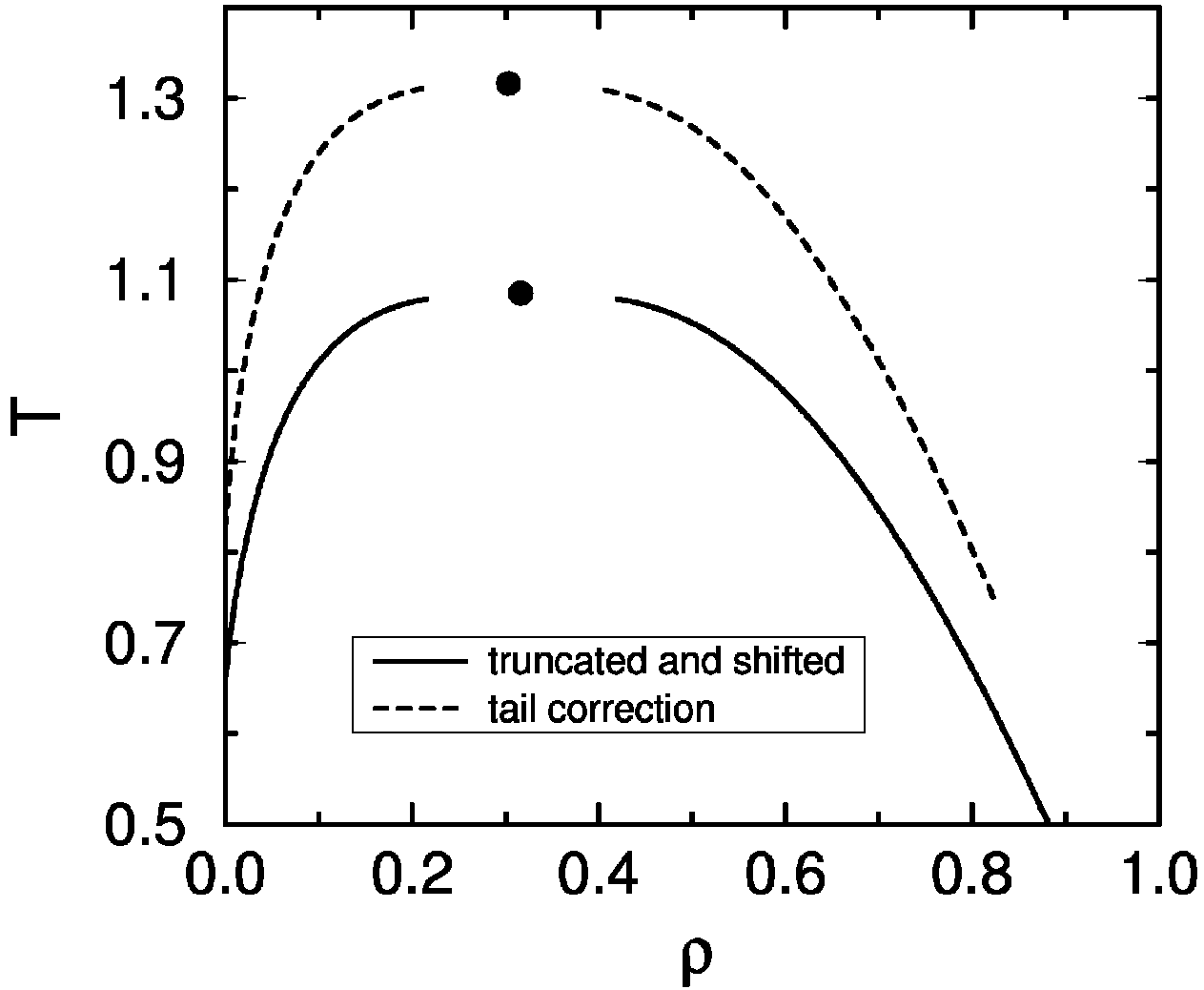
- The truncated and shifted Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) - u^{LJ}(r_c) & r \le r_c \\ 0 & r > r_c \end{cases}$$

Algorithm 5 (Calculation of the energies)

```
subroutine ener(x,en)                    determine the force
en=0                                     and energy
do i=1,npart
   f(i)=0                                set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                         loop over all pairs
     xr=x(i)-x(j)
     xr=xr-box*nint(xr/box)              periodic boundary conditions
     r2=xr**2
     if (r2.lt.rc2) then                 test cutoff
       r2i=1/r2
       r6i=r2i**3
       ff=48*r2i*r6i*(r6i-0.5)           Lennard-Jones potential
       f(i)=f(i)+ff*xr                   update force
       f(j)=f(j)-ff*xr
       en=en+4*r6i*(r6i-1)-ecut          update energy
     endif
  enddo
enddo
return
end
```

Phase diagrams of Lennard Jones fluids



Long ranged interactions

- Long-ranged forces require special techniques.
  - Coulomb interaction (1/r in 3D)
  - Dipolar interaction (1/r3 in 3D)

- ...and, in a different context:
  - Interactions through elastic stresses (1/r in 3D)
  - Hydrodynamic interactions (1/r in 3D)
  -

Reduced units

Example: Particles with mass **m** and pair potential:

$$v(r) = \epsilon f(r/\sigma)$$

Unit of length: σ

Unit of energy: ε

Unit of time: $\sigma\sqrt{m/\epsilon}$

Beyond standard MC

- **Non Boltzmann Sampling**
- **Parallel tempering**

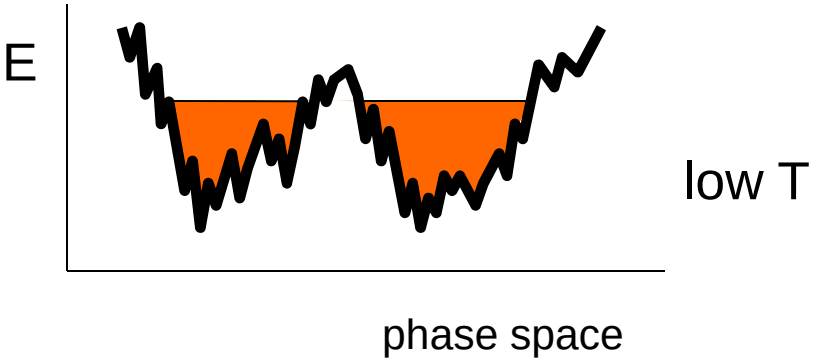
More to Come
on Thursday and Friday
(Daan Frenkel)

Parallel tempering/Replica Exchange

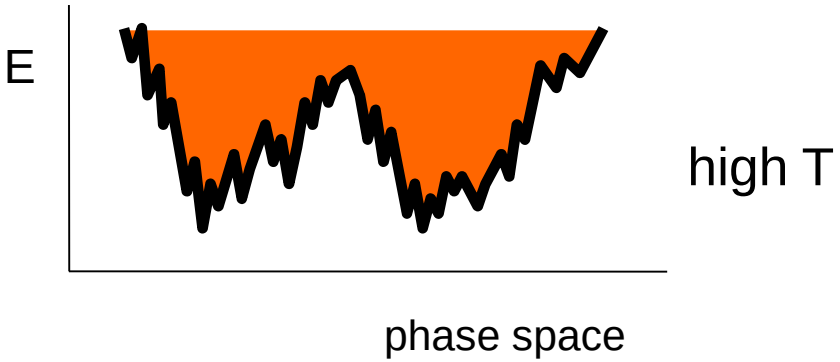
Ergodicity problems can occur, especially in glassy systems: biomolecules, molecular glasses, gels, etc.

The solution: go to high temperature

High barriers in energy landscape: difficult to sample



Barriers effectively low: easy to sample



Non-Boltzmann sampling

T_1 is arbitrary!

$$\langle A \rangle_{NVT_1} = \frac{1}{Q_{NVT_1}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right]$$

Why are we not using this?

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta_1 U(\mathbf{r}^N)\right]}$$

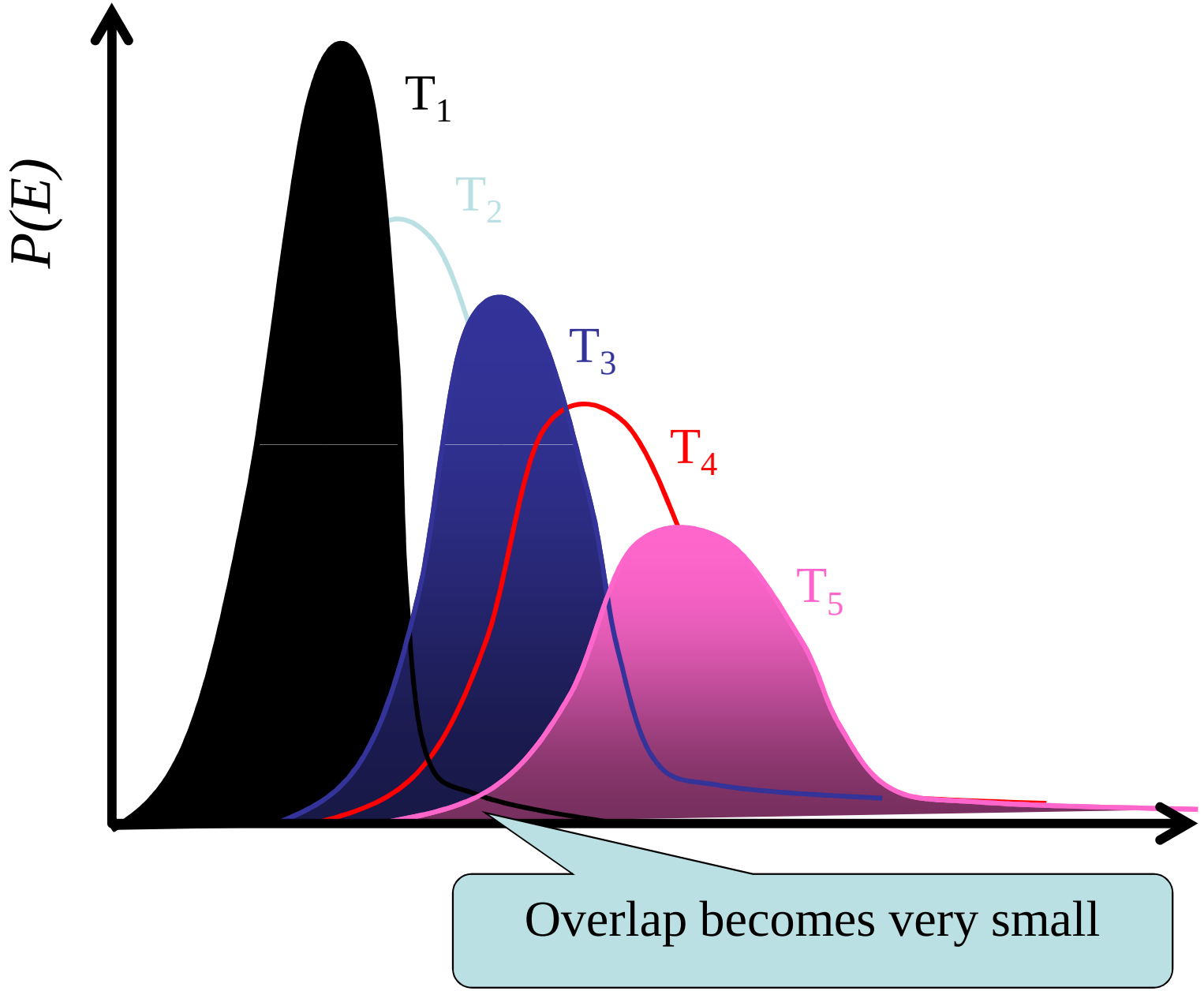
$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}$$

We only need a *single* simulation!

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_1 U(\mathbf{r}^N)\right] \exp\left[-\beta_2 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_1 U(\mathbf{r}^N)\right] \exp\left[-\beta_2 U(\mathbf{r}^N)\right]}$$

$$= \frac{\left\langle A \exp\left[(\beta_2 - \beta_1) U\right]\right\rangle_{NVT_2}}{\left\langle \exp\left[(\beta_2 - \beta_1) U\right]\right\rangle_{NVT_2}}$$

We perform a simulation at $T=T_2$ and we determine A at $T=T_1$


P(E)
T1
T2
T3
T4
T5
Overlap becomes very small

Parallel tempering/Replica Exchange

Simulate two systems simultaneously

system 1
temperature T_1

system 2
temperature T_2

$$e^{-\beta_1 U_1(r^N)}$$

$$e^{-\beta_2 U_2(r^N)}$$

total Boltzmann weight:

$$e^{-\beta_1 U_1(r^N)} e^{-\beta_2 U_2(r^N)}$$

Swap move

- Allow two systems to swap

system 2
temperature T_1

system 1
temperature T_2

$$e^{-\beta_1 U_2(r^N)}$$

$$e^{-\beta_2 U_1(r^N)}$$

total Boltzmann weight:

$$e^{-\beta_1 U_2(r^N)} e^{-\beta_2 U_1(r^N)}$$

$$\mathrm{acc}(1 \leftrightarrow 2) = \min\left(1, e^{(\beta_2 - \beta_1)[U_2(r^N) - U_1(r^N)]}\right)$$

Acceptance rule

The ratio of the new boltzmann factor over the old one is

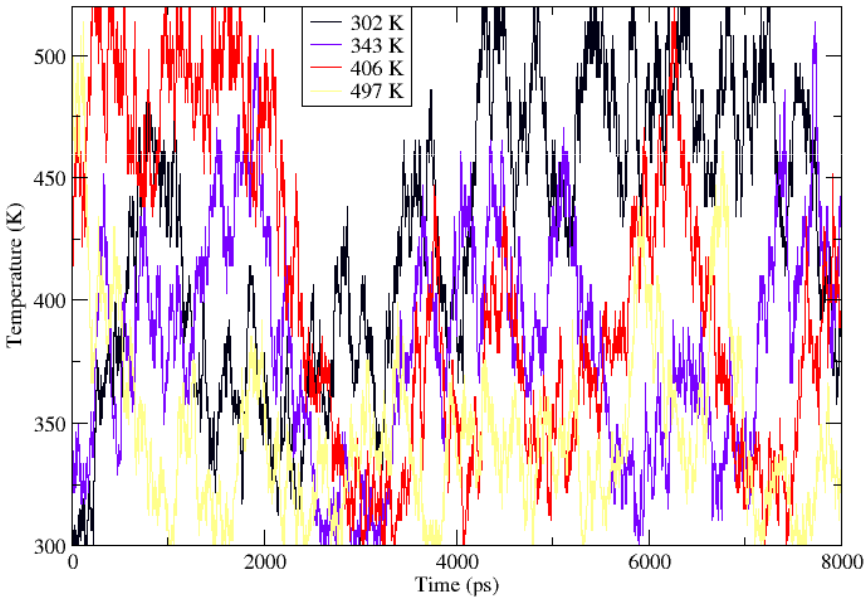
$$\frac{\mathcal{N}(n)}{\mathcal{N}(o)} = e^{(\beta_2 - \beta_1)[U_2(r^N) - U_2(r^N)]}$$

the swap acceptance ratio is

$$\mathrm{acc}(1 \leftrightarrow 2) = \min\left(1, e^{(\beta_2 - \beta_1)[U_2(r^N) - U_1(r^N)]}\right)$$

More replicas

Consider M replica's in the NVT ensemble at a different temperature.



A swap between two systems of different temperatures (T_i,T_j) is accepted if their potential energies are near.

other parameters can be used: Hamiltonian exchange

Questions

...

Lunch