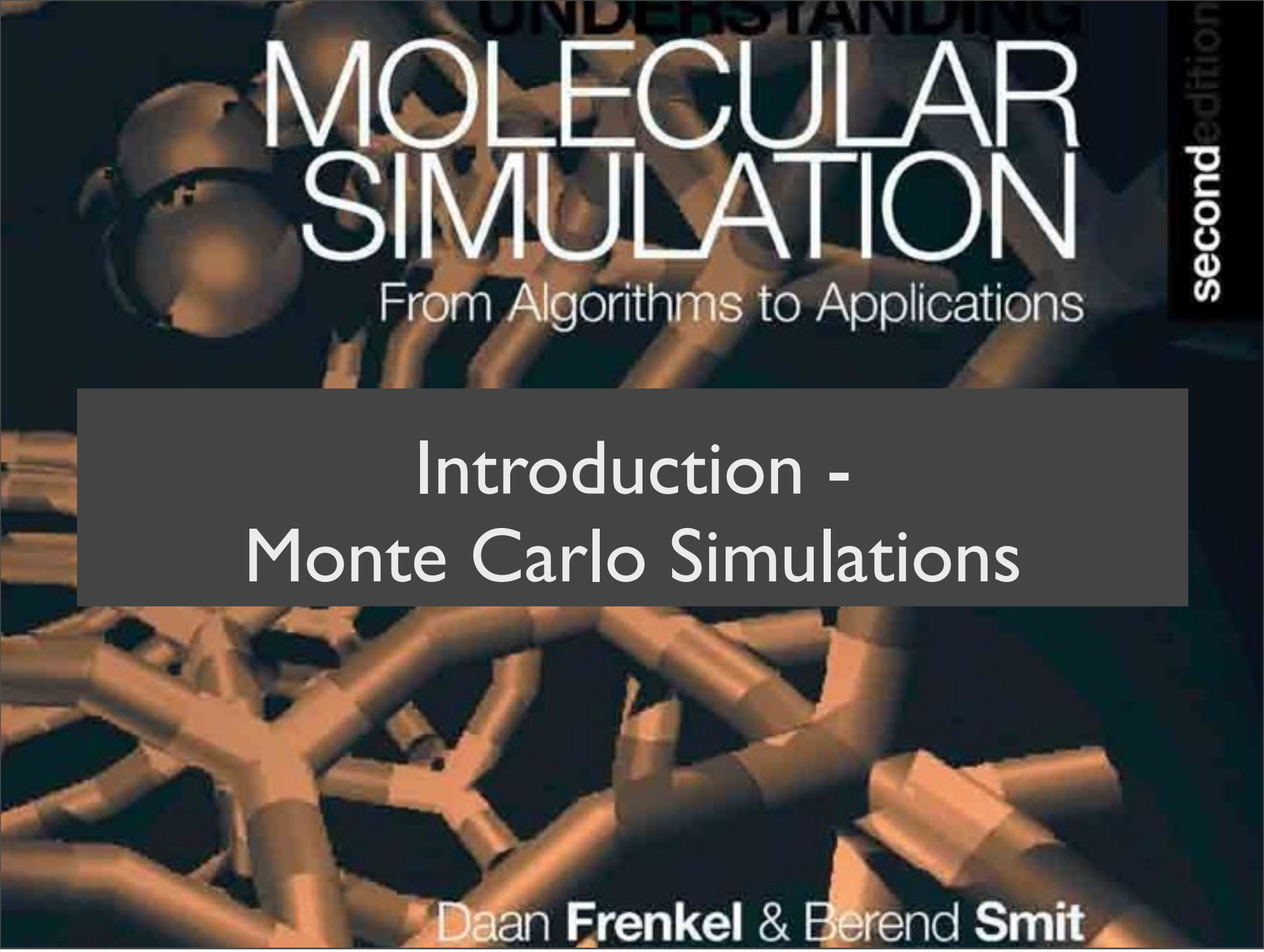
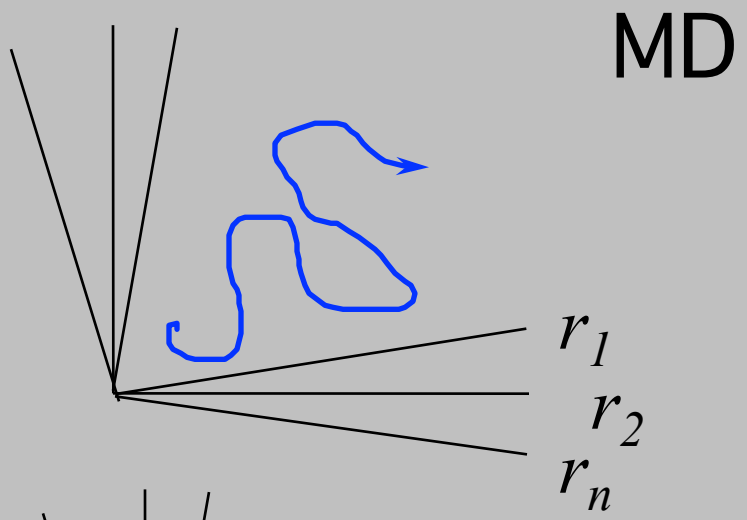
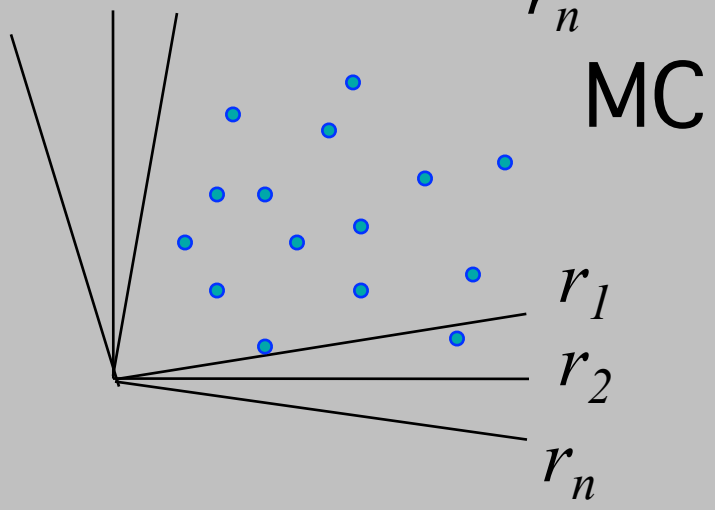

UNDERSTANDING
MOLECULAR SIMULATION
From Algorithms to Applications
second edition
Introduction -
Monte Carlo Simulations
Daan Frenkel & Berend Smit

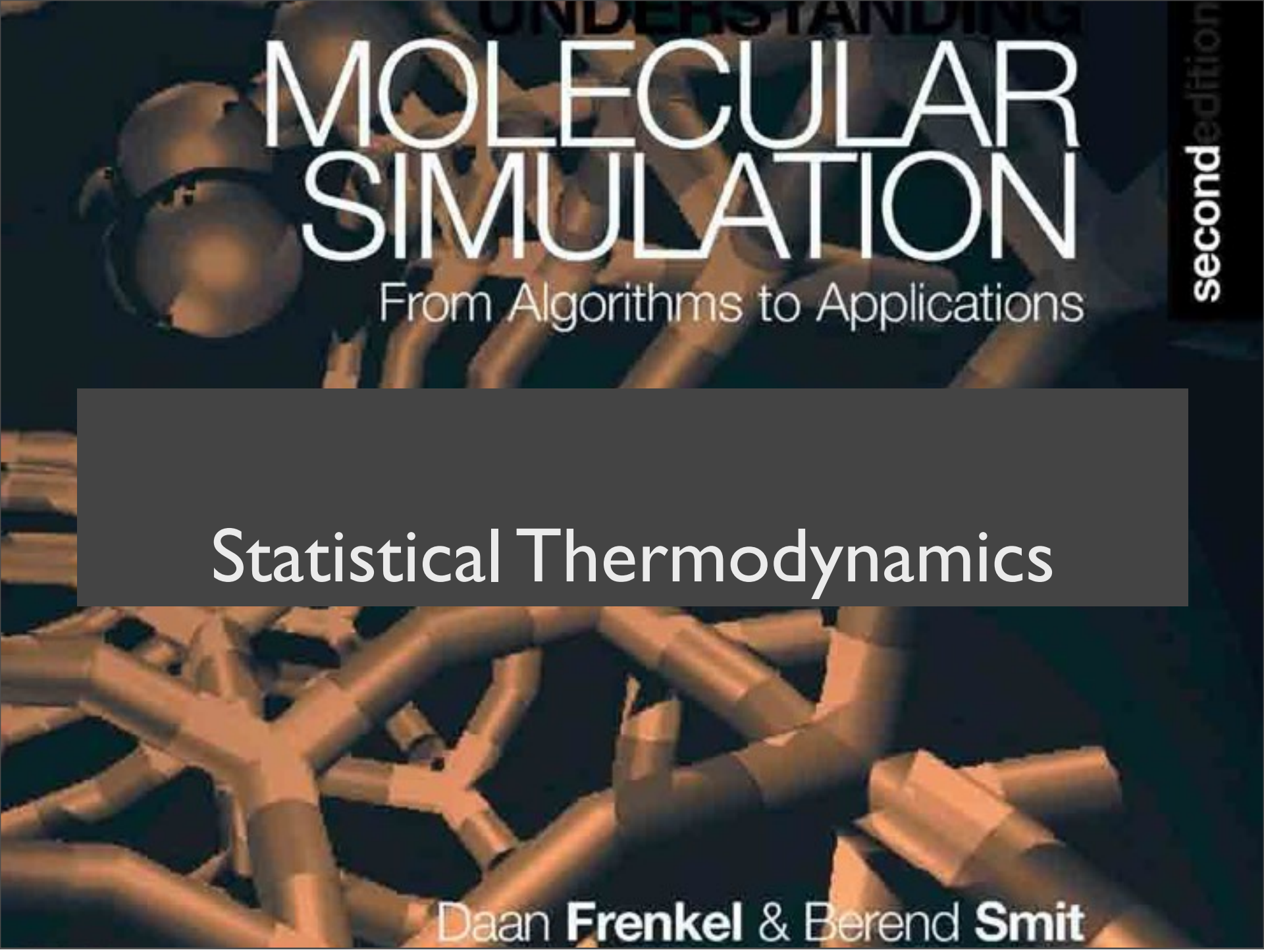
Molecular Simulations

- Molecular dynamics: solve equations of motion

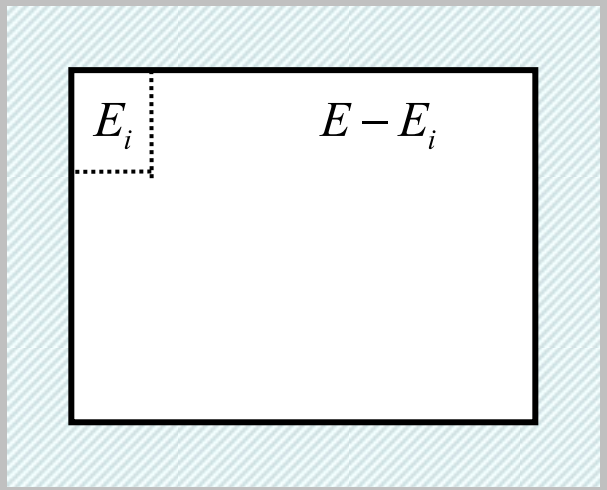


- Monte Carlo: importance sampling




UNDERSTANDING
MOLECULAR SIMULATION
From Algorithms to Applications
second edition
Statistical Thermodynamics
Daan Frenkel & Berend Smit

Canonical ensemble



$1/k_BT$

Consider a small system that can exchange heat with a big reservoir

$$\ln \Omega(E - E_i) = \ln \Omega(E) - \frac{\partial \ln \Omega}{\partial E} E_i + \cdots$$

$$\ln \frac{\Omega(E - E_i)}{\Omega(E)} = -\frac{E_i}{k_B T}$$

Hence, the probability to find E_i:

$$P(E_i) = \frac{\Omega(E - E_i)}{\sum_j \Omega(E - E_j)} = \frac{\exp(-E_i/k_B T)}{\sum_j \exp(-E_j/k_B T)}$$

$$P(E_i) \propto \exp(-E_i/k_B T)$$

Boltzmann distribution

Summary: Canonical ensemble (*N,V,T*)

Partition function:

$$Q(N,V,T) = \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U\left(r^N\right)\right]$$

Probability to find a particular configuration:

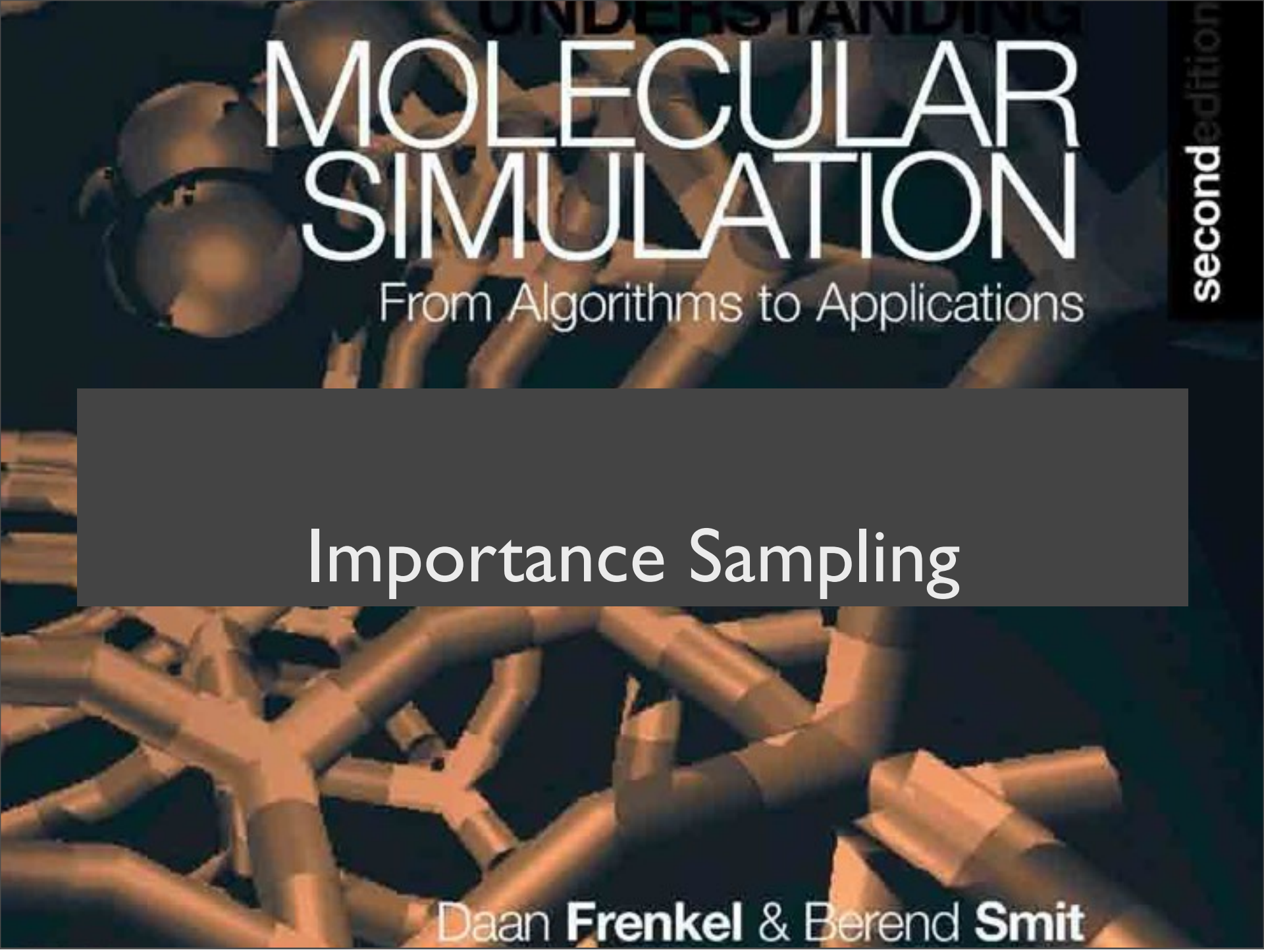
$$P(\Gamma) \propto \exp\left[-\beta U(\Gamma)\right]$$

Ensemble average:

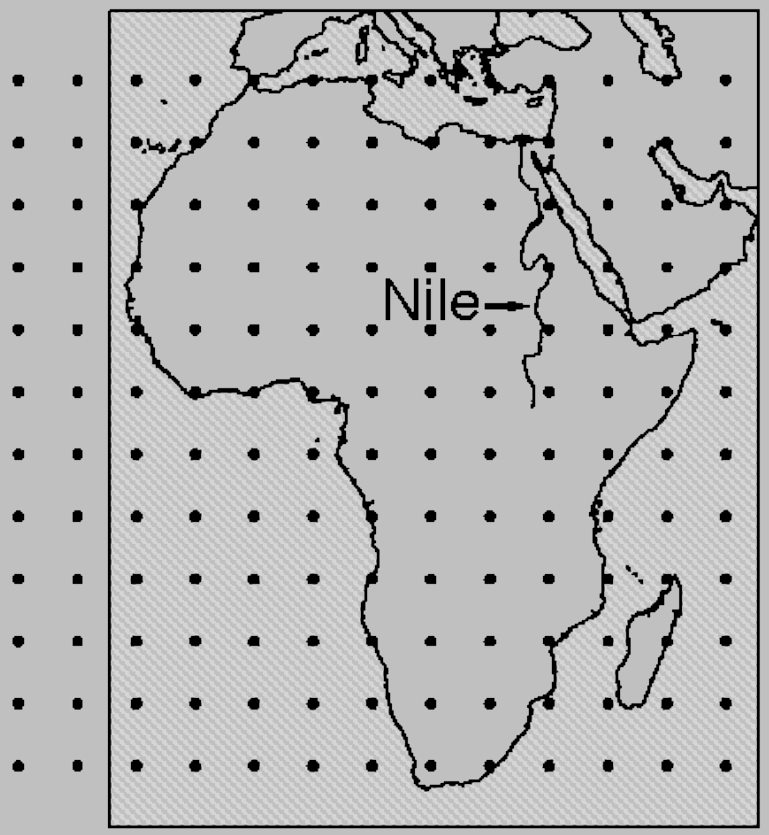
$$\langle A \rangle_{NVT} = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}^N A\left(\mathrm{r}^N\right) \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

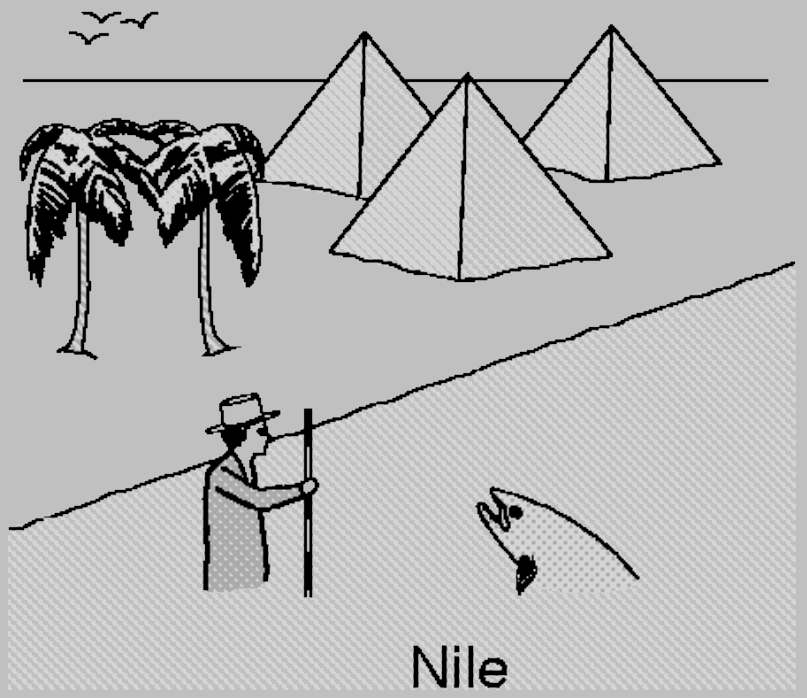
Free energy:

$$\beta F = -\ln Q_{N,V,T}$$


UNDERSTANDING
MOLECULAR SIMULATION
From Algorithms to Applications
second edition
Importance Sampling
Daan Frenkel & Berend Smit

Numerical Integration





Ensemble Average

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

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

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

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

Generate configurations using Monte Carlo moves

$$\left\{\mathbf{r}_1^N, \mathbf{r}_2^N, \mathbf{r}_3^N, \mathbf{r}_4^N \cdots, \mathbf{r}_M^N\right\} \qquad \bar{A} = \frac{1}{M} \sum_{i=1}^{M} A(\mathbf{r}_i^N) = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) P^{MC}(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N P^{MC}(\mathbf{r}^N)}$$

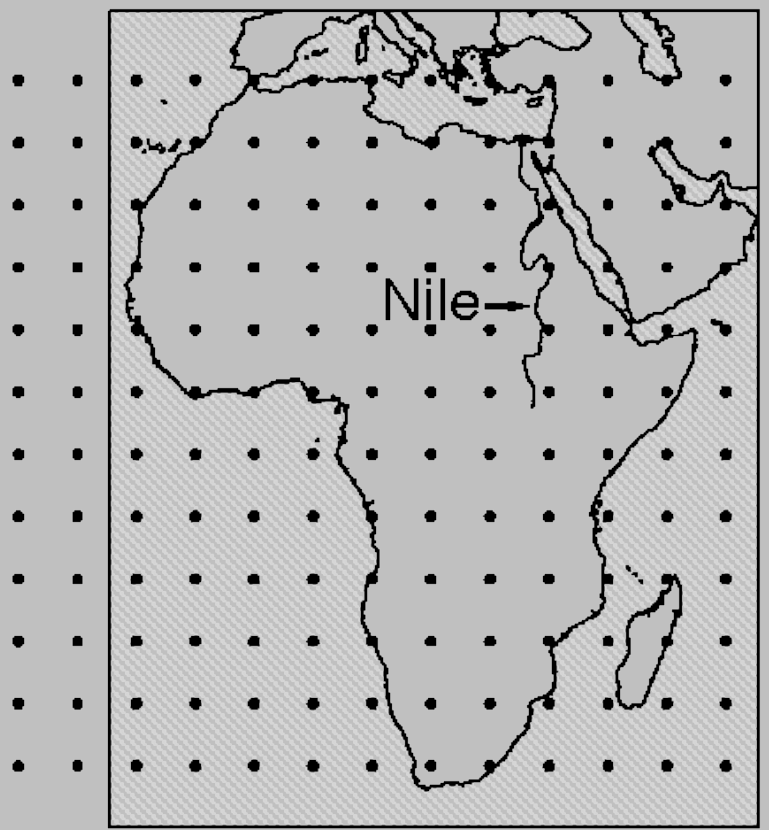
with:

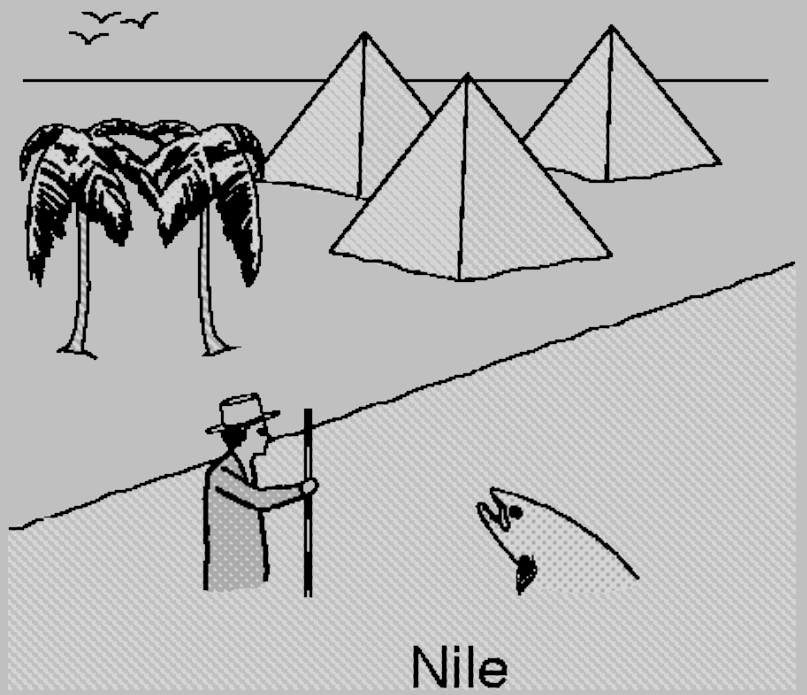
$$P^{MC}(\mathbf{r}^N) = C^{MC} \exp\left[-\beta U(\mathbf{r}^N)\right]$$

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

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

Importance Sampling





Algorithm 1 (Basic Metropolis Algorithm)

```
 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* `nsampth` *cycle.*

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* $[0, 1]$.

Questions

Desired distribution: NVT ensemble

- 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`?

Markov Processes

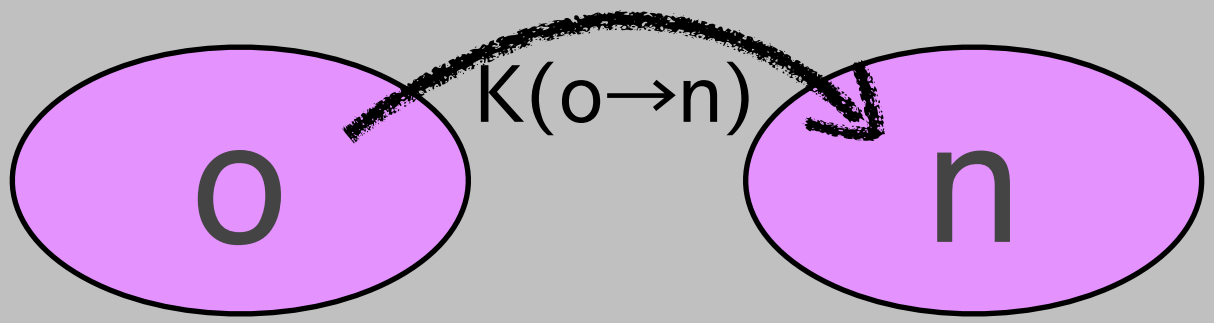
Markov Process

- Next step only depends on the current state
- Ergodic: all possible states can be reached by a set of single steps
- Detailed balance
- * Process will approach a limiting distribution

Ensemble - probability

- *P(o)*: probability to find the state *o*
- Ensemble: take a very large number (*M*) of identical systems: *N(o) = M x P(o)*; the total number of systems in the state o

Markov Processes - Detailed Balance

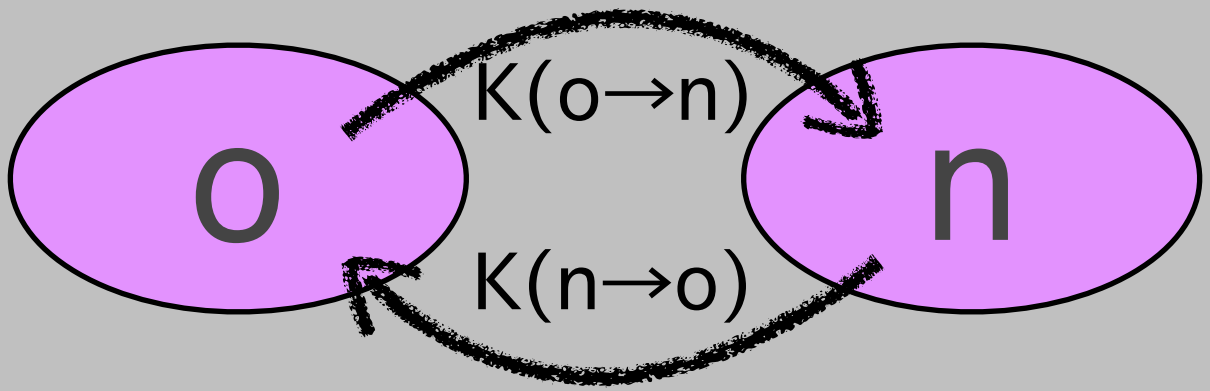


K(o → n): total number of systems in our ensemble that move *o* → *n*

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

- *N(o)* : total number of systems in our ensemble in state o
- *α(o → n)*: a priori probability to generate a move *o* → *n*
- *acc(o → n)*: probability to accept the move *o* → *n*

Markov Processes - Detailed Balance



Condition of detailed balance:

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

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

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

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

NVT-ensemble

In the canonical ensemble the number of configurations in state *n* is given by:

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

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n)}{N(o)}$$

Which gives as condition for the acceptance rule:

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

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* $[0, 1]$.

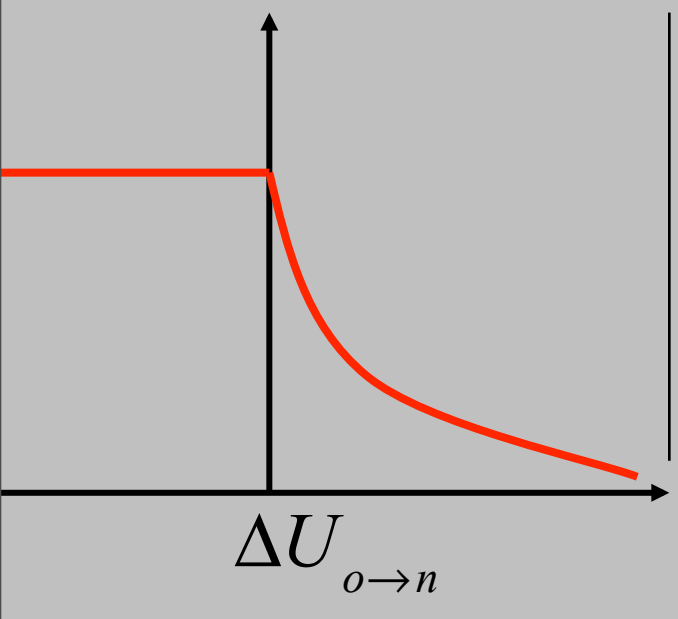
Metropolis et al.

Many acceptance rules that satisfy:

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

Metropolis *et al.* introduced:

$$\mathrm{acc}(o \to n) = \min\left(1, \exp\left[-\beta \Delta U_{o \to n}\right]\right)$$



If: $\Delta U_{o \to n} < 0$

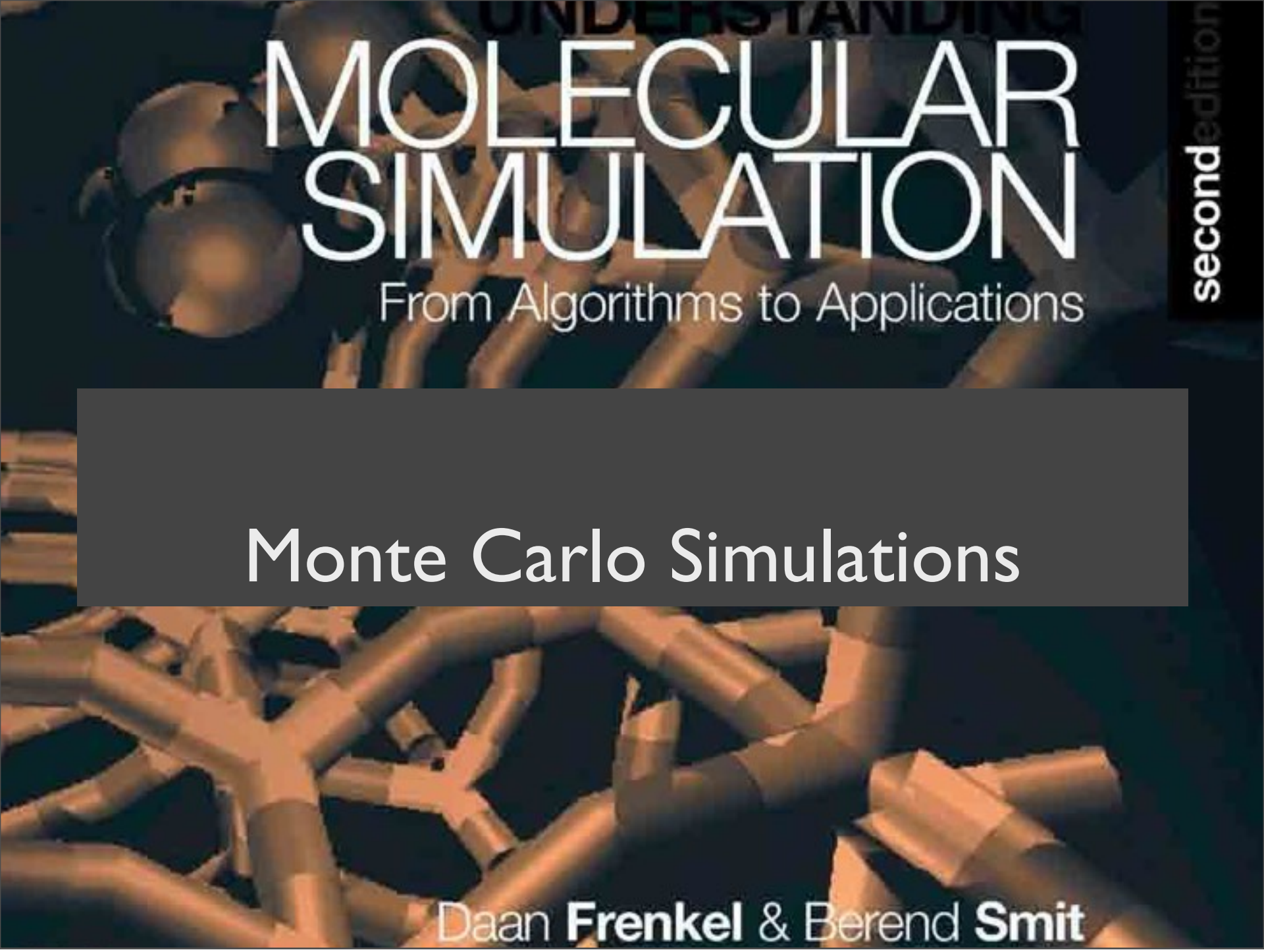
$$\mathrm{acc}(o \to n) = 1$$

$\Delta U_{o \to n} > 0$

$$\mathrm{acc}(o \to n) = \exp\left[-\beta \Delta U_{o \to n}\right]$$

Draw a uniform random number [0;1] and accept the new configuration if:

$$\mathrm{ranf} < \exp\left[-\beta \Delta U_{o \to n}\right]$$


UNDERSTANDING
MOLECULAR SIMULATION
From Algorithms to Applications
second edition
Monte Carlo Simulations
Daan Frenkel & Berend Smit

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`?

Detailed Balance

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* $[0, 1]$.

Mathematical

Transition probability from *o* → *n:*

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

As by definition we make a transition:

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

The probability we do not make a move:

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

This term is ≠ 0

Model

Let us take a spin system

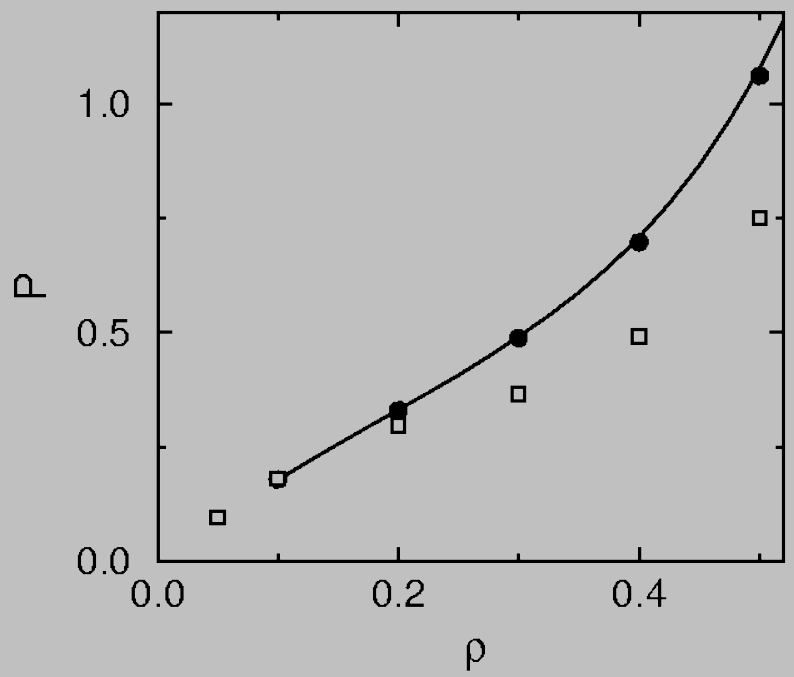
With energy $U\uparrow = +1$ and $U\downarrow = -1$

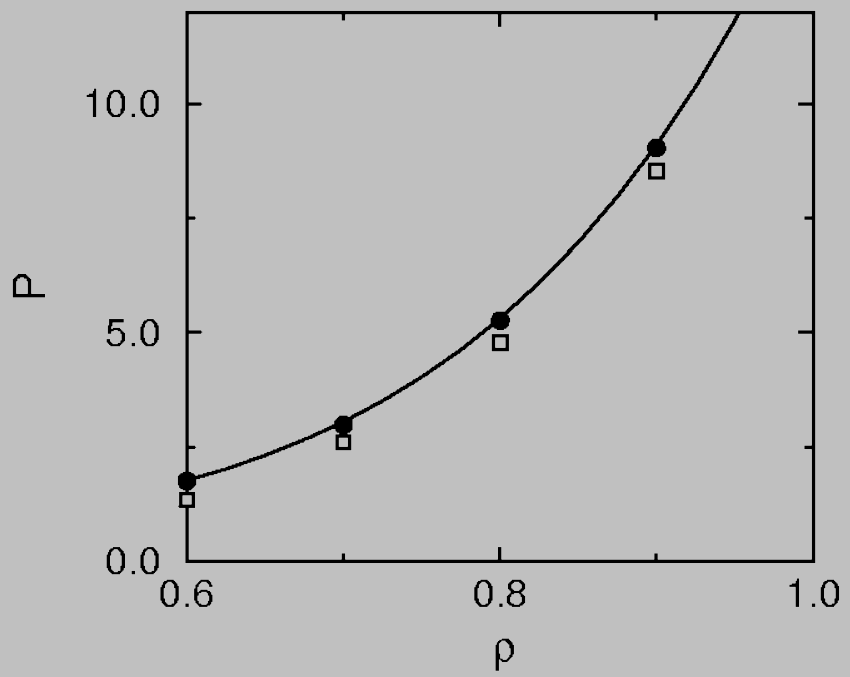
$$P(\uparrow) \propto \exp\left(-\frac{u\uparrow}{k_B T}\right)$$

If we do not keep the old configuration:

Independent of the temperature

Lennard Jones fluid

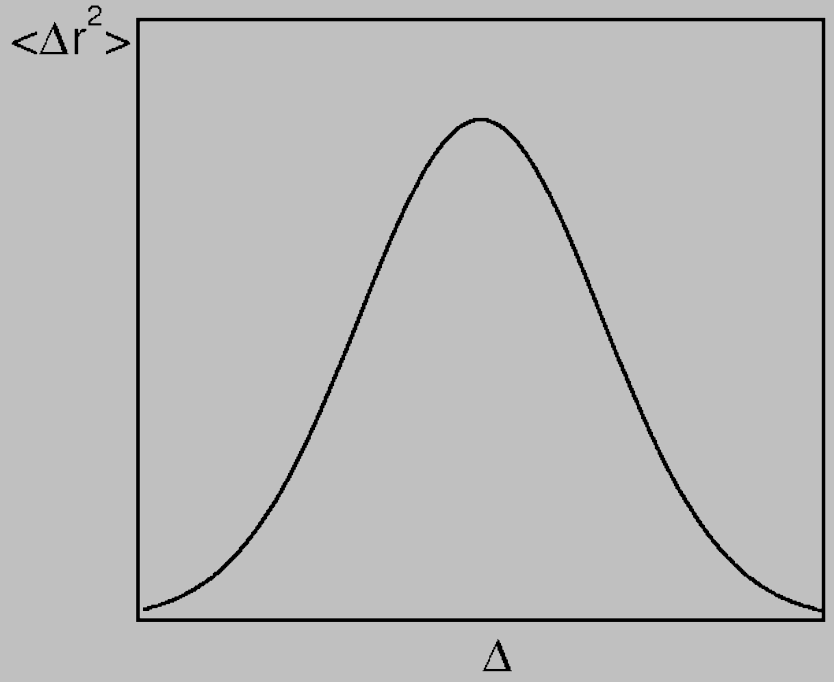


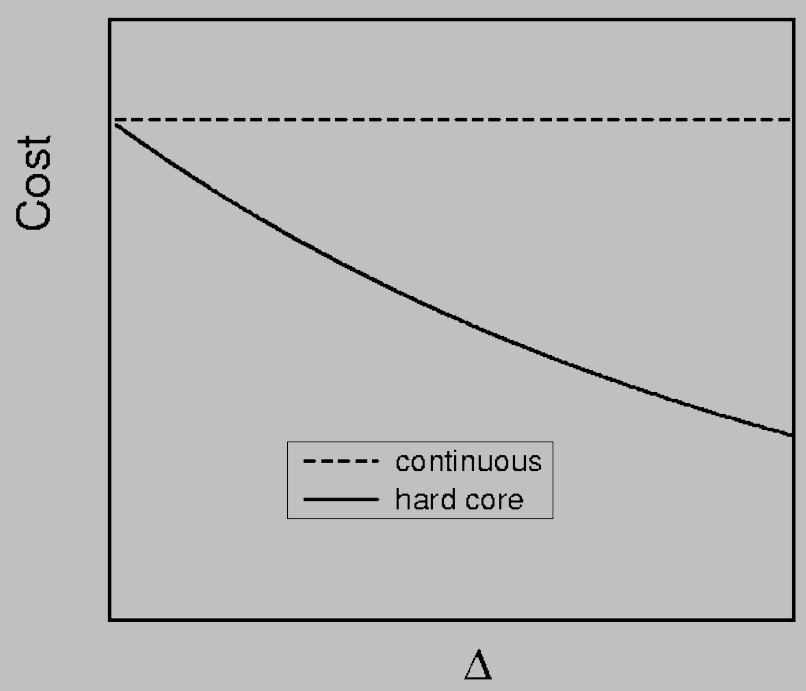


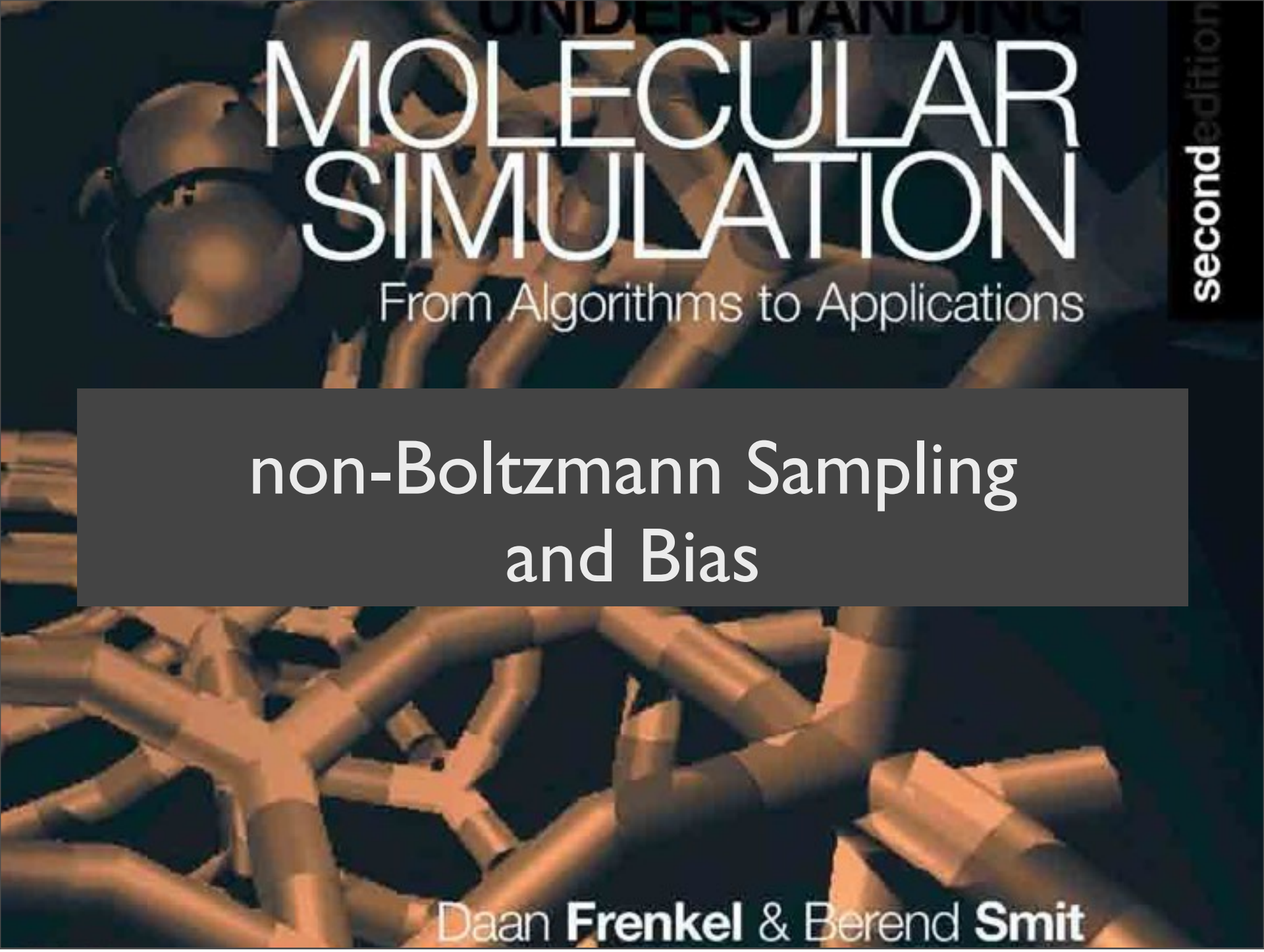
Questions

- 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`?**

Not too big Not too small







non-Boltzmann Sampling and Bias

Non-Boltzmann sampling

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

T_1 is arbitrary

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

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

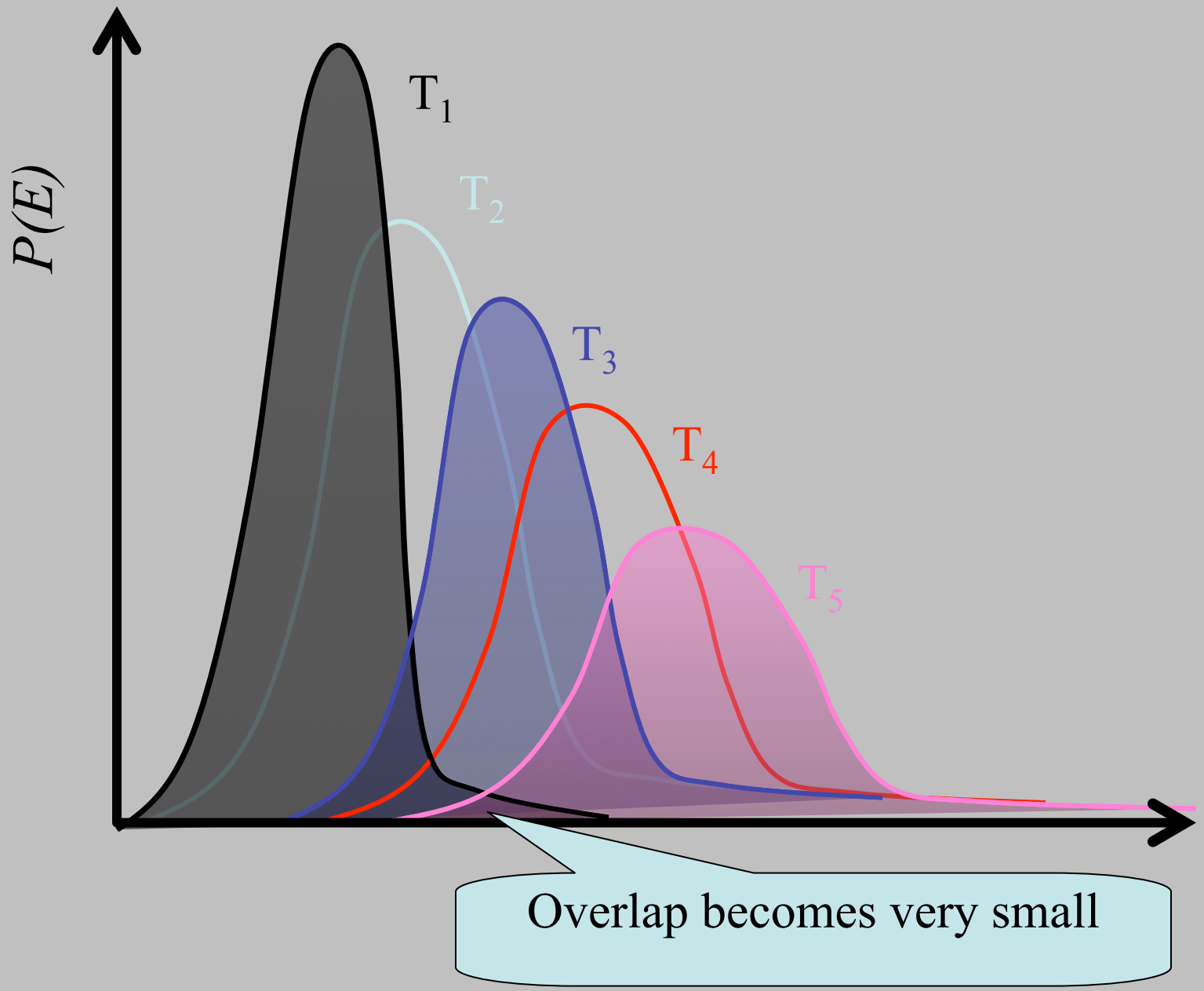
$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}$$

Why are we not using this?

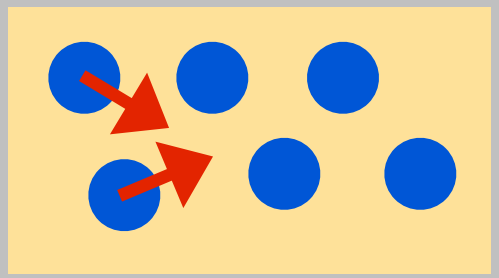
$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_1 U(\mathbf{r}^N)\right] \exp\left[-\beta_2 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_1 U(\mathbf{r}^N)\right] \exp\left[-\beta_2 U(\mathbf{r}^N)\right]}$$

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

We only need a **single** simulations


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

Parallel Monte Carlo



How to do a Monte Carlo simulation in parallel?

- (trivial but works best) Use an ensemble of systems with different seeds for the random number generator
- Is it possible to do Monte Carlo in parallel?
  - Monte Carlo is sequential!
  - We first have to know the fait of the current move before we can continue!

Parallel Monte Carlo - algorithm

Naive (and wrong)

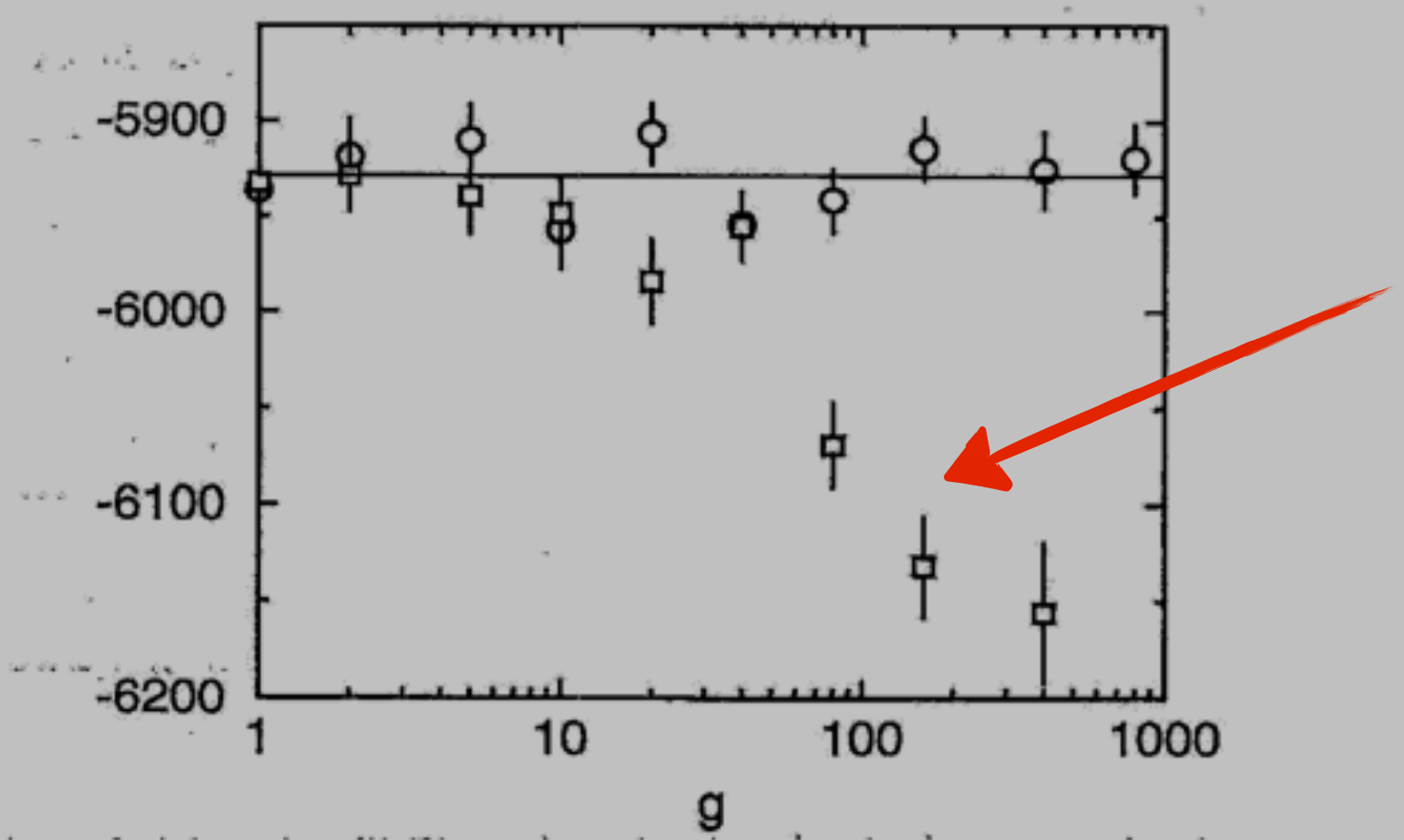
1. Generate k trial configurations in parallel
2. Select out of these the one with the lowest energy

$$P(n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$

3. Accept and reject using normal Monte Carlo rule:

$$\mathrm{acc}\left(o \to n\right) = \exp\left[-\beta\left(U_n - U_o\right)\right]$$

Conventional acceptance rules



The conventional acceptance rules give a bias

What went wrong?

Detailed balance!

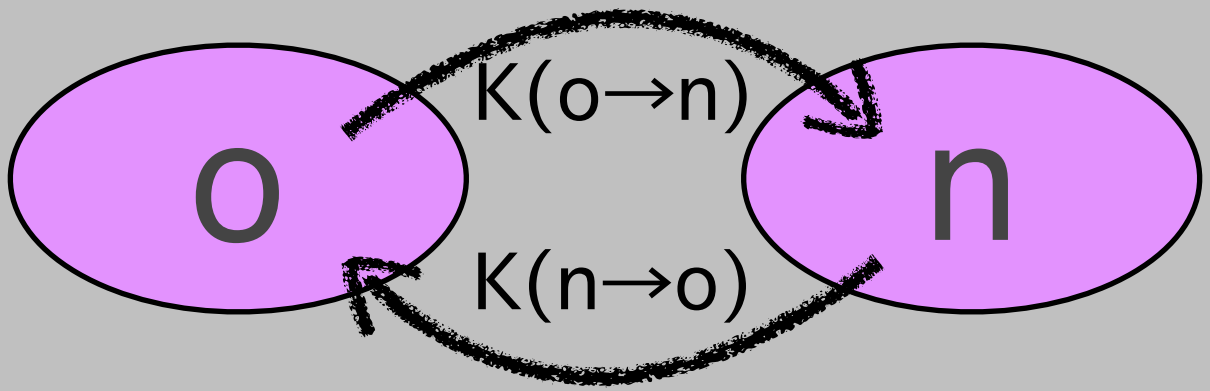
$$K(o \to n) = K(n \to o)$$

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

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

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

Markov Processes - Detailed Balance



Condition of detailed balance:

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

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

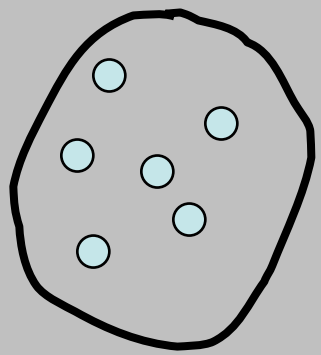
$$K(n \to o) = N(n) \times \alpha(n \to o) \times \mathrm{acc}(n \to o)$$

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

?

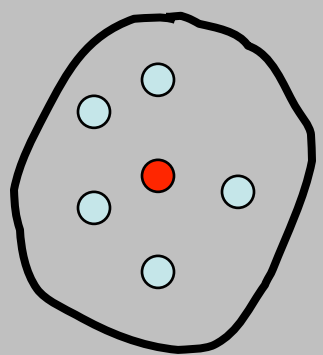
$$K(o \to n) = N(o) \times \alpha(o \to n) \times \text{acc}(o \to n)$$

$$\alpha(o \to n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$



$$\alpha(o \to n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)}$$

$$\alpha(n \to o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$



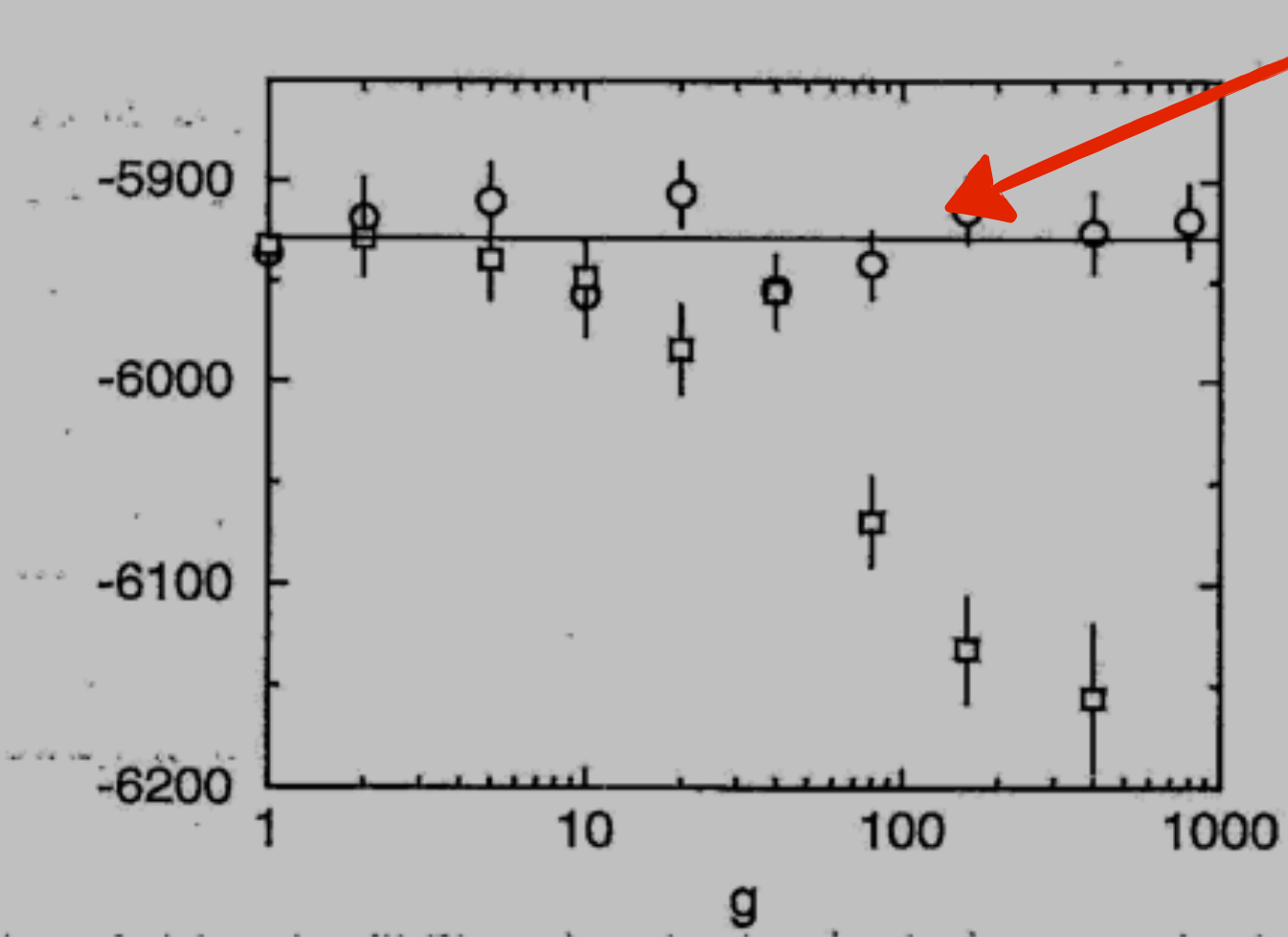
$$\alpha(n \to o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}$$

$$\frac{\text{acc}(o \to n)}{\text{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

$$\alpha(o \to n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)} \qquad \alpha(n \to o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}$$

$$\frac{\text{acc}(o \to n)}{\text{acc}(n \to o)} = \frac{N(n) \times \dfrac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}}{N(o) \times \dfrac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)}} = \frac{W(n)}{W(o)}$$

Conventional acceptance rules



Modified acceptance rules remove the bias exactly