Exercise 10 The n-fold way

In the previous exercise you have probably found that a large part of the test configurations will be rejected and a large part of the CPU time is wasted by unsuccessful cycles.

There are ways to make this less problematic. In off-lattice codes, where molecules are not confined to lattice positions but can have all possible (x, y, z), the step size of the trial moves is often adjusted to obtain an acceptable acceptance percentage.

For lattice models, like the one we use here, a possibility is to use the n-fold way by Bortz, Kalos and Lebowitz [1]. This methods requires that all possible trial configurations from a certain configuration are known. In our lattice model this is the case and we can therefore use this method. The algorithm is now transformed into

Step 1: Determine all possible step configurations w.r.t. the current step with one growth unit change

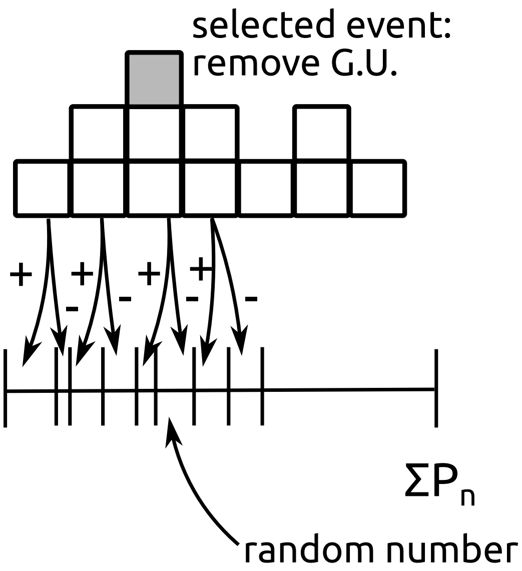
Step 2: Determine all transition probabilities (see figure below) and make a cumulative series

Step 3: Pick a random number between 0 and $\sum_i P_{0\to1,i}$

Step 4: By comparing this random number to the cumulative probabilities determine which event will occur (see figure below)

Step 5: Move to the configuration chosen in step 4

Step 6: Return to step 1



This algorithm is computationally more expensive per cycle, since it requires to determine the probabilities of all possible transitions. However, when the acceptance percentage is low in the Metropolis algorithm this will compensated by the smaller amount of cycles that is needed to achieve a good sampling. Furthermore, not all transition probabilities need updating each cycle. By a clever bookkeeping schemes the computational load can be kept low.

a) How many different transitions are possible and what are their probabilities?

b) If one would change the step at one position, how many probabilities change for the next iteration?

c) Write a new Matlab function `nfoldMCmoves` that simulates a step like in the previous exercise and allows to calculate the kink density and step energy. A few hints: make a vector with that contains all probabilities and during the simulation only update those probabilities that change.

d) Check that both routines give the same result.

Exercise 11 Kinetic Monte Carlo

This exercise will extend the previous scripts to situations outside equilibrium.

The n-fold way was originally designed to perform kinetic Monte Carlo. We will convert nfoldMCmoves into kMCmoves.

a) Name two reasons why we can convert the ensemble-averaging Monte Carlo into kinetic Monte Carlo.

b) Create `kMCmoves` from `nfoldMCmoves` and replace all probabilities by rates. For this we use the following routine:

$$k^{+} = \frac{kT}{h} \exp\left(\frac{\Delta\mu}{kT}\right) \tag{1}$$

$$k^{-} = \frac{kT}{h} \exp\left(-\frac{\Delta E}{kT}\right). \tag{2}$$

The prefactor $\frac{kT}{h}$ describes the attempt frequency and is usually of the order 10^{12} s^{-1}. The rate of incoming particles is determined by the difference in chemical potential between the gas phase and the crystal, *i.e.*, the driving force for crystallisation. The desorption rate is determined by the binding energy. These rates also fulfill microscopic reversibility, which is necessary to obtain no drifts in the simulations.

c) Add the time in these simulations and a function to calculate the step velocity:

$$v_{\mathrm{step}} = \frac{N_{\mathrm{grow}} - N_{\mathrm{etch}}}{t} \tag{3}$$

with N_{grow} and N_{etch} the number of grow and etch events, respectively.

d) Plot the step velocity and kink density as a function of $\Delta\mu/kT$.

e) In kinetic Monte Carlo the rates make a difference in the final results. Check this by adding the same additional energy to the barrier of the process.

References

[1] A. B. Bortz, M. H. Kalos and J. L. Lebowitz, J. Comp. Phys. 17 (1975) 10.