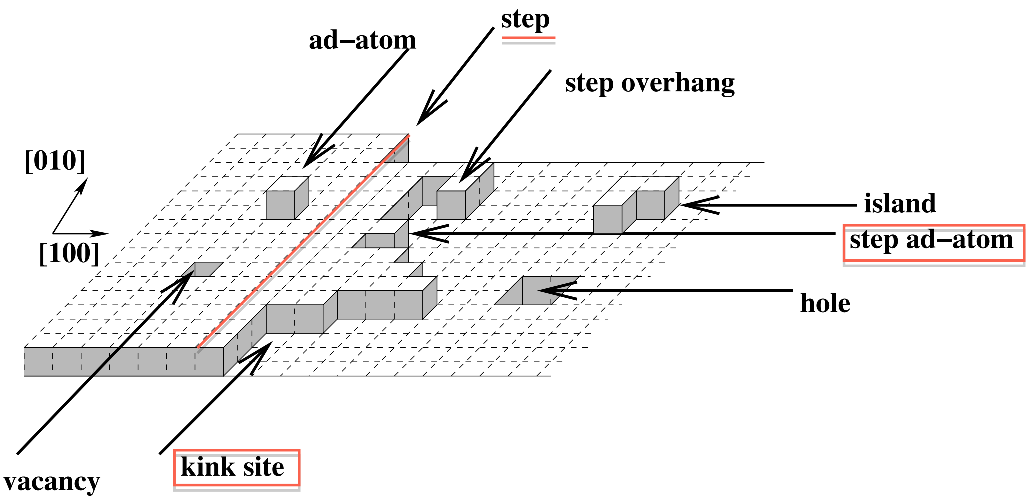
Metropolis Monte Carlo simulation of surface steps

We will write an algorithm that simulates the step edge on a crystal surface in thermodynamic equilibrium using the Monte Carlo technique. Because we are in equilibrium we will start by Metropolis Monte Carlo.

We will study the Kossel (100) surface and a step in the [010] direction. The Kossel crystal is a simple cubic crystal with one molecule per lattice cell and a binding of value ϕ between neighbouring cells in all three directions. At $T > 0$ such a step will not be perfectly straight but have several kinks. The number of kinks, or the kink density, will strongly depend on the temperature and the bond strength between the particles. The kink patterns can be come quite complex at high temperatures. It is even possible to have overhangs for instance. Here we will restrict ourselves to a Solid-On-Solid model in the step direction. This means that particles can only attach to the step when it will have a neighbour in the [100] direction (see figure below) or that particles can desorb from the step when it leaves no other particle unattached in the [100] direction. This means that we *only* consider kink sites and step ad-atoms as depicted in the figure below. 'Fingers' of multiple step ad-atoms will off course be possible.



The advantage of this simple model is that we can describe the step as one dimensional vector (or 1xn matrix in Matlab) consisting of integers. These integers represent the height of the step excursions. The step in the figure will then be represented by vector $X_0 = (1, 1, 1, 3, 6, 5, 4, 2, 3, 2, 2, 2, 4, 0)$ if we neglect the step overhang. An infinitely long step can be simulated by applying periodic boundary conditions in the direction of the step.

We will determine properties of the step using a Monte Carlo algorithm. Starting with initial step configuration X_0 the algorithm follows the following steps:

Step 1: Choose a random trial configuration X_1 w.r.t. X_0

Step 2: Determine the energy difference between X_0 and X_1

Step 3: Accept or reject X_1 according to the Metropolis algorithm

Step 4: Return to step 1

The code below is the heart of the Metropolis Monte Carlo simulation. In the following exercises you will gradually fill the spaces `...input...`.

(pseudocode)

```
function st = MetropolisMCmoves(...input...)
st = step;
E = calculateE(step);
for I = 1:moves
    trial = step;
    trial = ...input....;
    Etrial = calculateE(trial);
    DeltaE = Etrial - E;
    if (DeltaE < 0) %decide whether trial is accepted
      ...input...
    else
      ...input...
    end

    if (accepted)
      st = trial;
      E = Etrial;
      AccMoves = AccMoves + 1;
    end

    if (accepted && mod(Accmoves,length(st)) == 0)
      %sample the properties of the step and average
      ...input...
    end
end
```

Exercise 5.1 Metropolis Monte Carlo

a) Write a function that calculates the total energy of the step with respect to the ground state (a perfectly straight step of the same number of atoms).

b) Create a method to generate a trial configuration that differs one particle from the previous configuration. Make sure that all possible configurations are generated with equal probability.

c) Add the Metropolis algorithm to determine whether a trial configuration will be accepted or rejected.

d) Write functions that analyse the properties of the step, *e.g.*, the kink density (the number of kinks per unit length along the step direction) and the step energy (the number of broken bonds per unit length along the step direction). Take the average of these properties during the simulation. The properties of the step are not sampled each iteration since the configurations X_n and X_{N+1} are correlated. After all, they only differ by one molecule. The system needs to be considerable changed before the properties are calculated again. Furthermore, if you start with a straight step, the step needs to equilibrate to its equilibrium configuration before you start sampling.

e) Run a number of simulations for different temperatures and watch how the kink density and step energy converges for increasing Monte Carlo cycles. Determine the percentage of accepted trial configurations.

Exercise 5.2 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 configuration 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 configurations $X_{1,i}$ w.r.t. X_0

Step 2: Determine all energy differences between X_0 and $X_{1,i}$

Step 3: Determine the transition probabilities from X_0 to all $X_{1,i}$ ($P_{0\to1,i}$)

Step 4: Pick a random number between 0 and $\sum_i P_{0\to1,i}$

Step 5: By comparing this random number to the probabilities determine which event will occur

Step 6: Move to the configuration chosen in step 5

Step 7: 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? If one would add one particle to the step, how many probabilities change for the next iterations?

b) Write a new 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.

c) Check that both routines give the same result.

Exercise 5.3 Sampling and microscopic reversibility

According to microscopic reversibility or detailed balance, the transition probability from state i to state f should fulfill the following criterion:

$$\frac{P_{i\to f}}{P_{f\to i}} = \exp\left(-\frac{E_f - E_i}{kT}\right). \tag{1}$$

(P is T in the lecture notes) From my slide 9, eq. 1:
T(i⟶f)/T(f⟶i)=p(f)/p(i)=exp(-Ef/kT)/exp(-Ei/kT)
(sufficient but not necessary condition)

The Metropolis probability scheme fulfills this requirement. Other probability scheme do as well. The final result should be independent of the probability scheme; different schemes do however result in different convergence behaviour: less/more cycles are needed before the properties converge to their equilibrium value.

a) Implement the following probability scheme:

$$P_{i\to f} \quad = \quad \exp\left(-\frac{E_f - E_i}{2kT}\right) \tag{2}$$

Come interpretare l'eq (2)? essendo P_{i\to f} una probabilità, non dovrebbe eccedere 1, e quindi dovrebbe essere completata con un limite superiore. Se la completiamo al solito modo "alla Metropolis", cioè P_{i=>f}=min[1,exp(-(Ef-Ei)/2kT)], questo equivale a Metropolis con 2T.
Se invece consideriamo che in (2) rispetto a (1) manca il denominatore, e che P_{f=>i} =exp(-(Ei-Ef)/2kT) , risulta che P_{i=>f}/P_{f=>i} = exp(-(Ef-Ei)/kT), cioe' esattamente come (1), per T.

Check that this results in the same properties and check the convergence.

b) Do the same for:

$$P^{+} \quad = \quad 1 \tag{3}$$

$$P^{-} \quad = \quad \exp\left(-\frac{E_f - E_i}{kT}\right), \tag{4}$$

Qui il sistema non conserva il numero di particelle. Anche qui l'algoritmo ha una certa arbitrarietà:
P+ e P- si possono riferire alle probabilità di accettare o meno una mossa dopo aver scelto (in modo random) se aggiungere o rimuovere un atomo da un dato sito
Oppure possono essere proprio le probabilità di aggiungere (sempre) o rimuovere (con una certa probabilità) un atomo in un dato MC step.

where P^{+} represents the transitions that lead to addition of particles and P^{-} represent transitions that lead to the removal of particle.

References

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