LECTURE 11: Monte Carlo Methods III

November 21, 2011

In this last chapter, we discuss non-equilibrium Monte Carlo methods. We concentrate on lattice systems and discuss ways of simulating phenomena such as surface diffusion and crystal growth.

1 Models of surface diffusion and growth

1.1 Atomic processes in surface growth

Crystalline materials are often grown using methods where new material is deposited on the surface of a substrate crystal. The growth is governed by a collection of *atomic scale* processes. The three most important processes are the *deposition* of new atoms, *desorption* of surface atoms and *diffusion* of atoms on the surface.

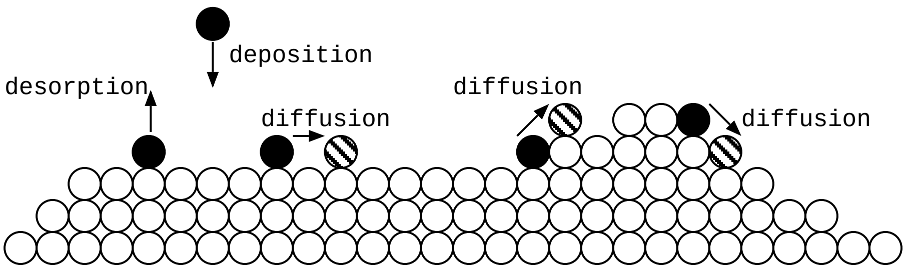

Figure 1: Atomic processes during surface growth.

In the following, we will consider lattice models which can be used to describe these processes and to simulate the growth process.

1.2 The lattice gas model

We will first consider the simplest model of *surface diffusion*. Suppose that we have a collection of moving particles on a surface. In the *lattice gas model*, each particle occupies a given lattice site. The surface is thus modeled by a lattice where each site is marked as either *occupied* ($n_i = 1$) or *unoccupied* ($n_i = 0$) by a particle.

The **total number of particles** is given by

$$N_p = \sum_{i=1}^{N} n_i$$

where N is the total number of sites on the lattice (e.g. for a square two-dimensional lattice $N = L \times L$).

The **particle concentration** (or surface coverage) is defined as

$$\theta = \frac{N_p}{N}$$

This is the fraction of occupied sites on the lattice.

Note that the lattice gas model is another interpretation of the **Ising model**. (Occupied sites correspond to up spins and unoccupied to down spins).

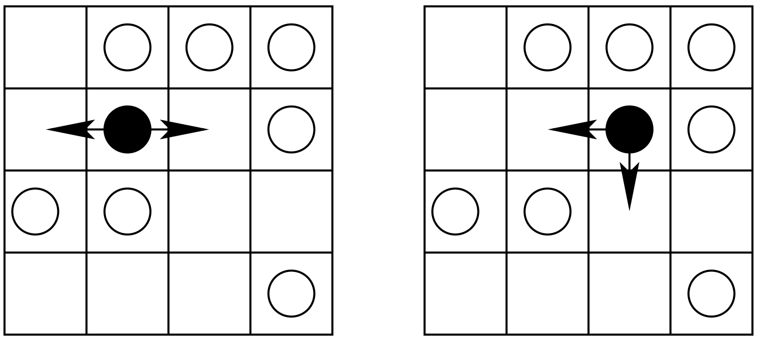

Figure 2: Lattice gas model of surface diffusion. At time t_1 (left figure) the black particle is chosen to jump. The jump can be directed either to the left or right (to unoccupied neighbouring sites). At some later time $t_2 > t_1$, the black particle is chosen again. The arrows point the two possible jump directions this time.

1.3 Diffusion model

In the simplest approach, the atoms are not mutually interacting. The only limitation is that each lattice site can be occupied by at most one atom at a time. The surface diffusion is modeled by considering each atom as a *random walker*. The random walks are *correlated* because each walker is limited by the presence of the other atoms on the surface.

Diffusion consists of random jumps of atoms to neighbouring unoccupied sites. If all the neighbouring sites are occupied by other atoms, then nothing happens (but the walks do not die if they cannot perform a jump).

If the concentration of atoms is very low, then the probability of encountering an occupied neighbouring site is very small and the random walks are virtually uncorrelated. In this case, the mean squared displacement of the random walkers follows the formula for a simple random walk (RW)

$$\langle [\mathbf{r}(t) - \mathbf{r}(0)]^2 \rangle \sim t$$

Here $\mathbf{r}(t)$ denotes the position of the random walker at time t. (The walk begins at $t = 0$).

When the concentration increases, it becomes more difficult for the atoms to find unoccupied neighbouring sites. Consequently, the mean squared displacement of the atoms will decrease with respect to the low concentration case.

The *diffusion process* can be described using the equation

$$\frac{\partial P(\mathbf{r},t)}{\partial t} = D\nabla^2 P(\mathbf{r},t)$$

where $P(\mathbf{r},t)$ is the probability that an atom is at the position $\mathbf{r}$ at time t. D is a diffusion constant which measures the mobility of the atoms.

The **diffusion constant** can be determined using the Einstein relation

$$D = \lim_{t\to\infty} \frac{1}{2dt} \langle [\mathbf{r}(t) - \mathbf{r}(0)]^2 \rangle$$

where d is the dimension of the movement.

For the lattice gas model, the diffusion constant depends on the concentration of atoms on the surface.

Monte Carlo algorithm

The system is composed of a square $L \times L$ lattice. The given concentration of atoms $\theta = N_p/(L \times L)$ is initially placed on random sites of the lattice. The simulation algorithm is as follows:

1. Create the initial configuration by choosing the occupied lattice sites randomly.
2. Choose the index of the moving atom randomly ($i \in [0, N_p - 1]$).
3. Choose the direction of the jump randomly (left, right, up or down).
4. If the chosen site is empty, move the atom there. Else do nothing.
5. Update quantities of interest.
6. Go to 2.

Implementation

1. Input parameters (examples in parenthesis)

```
L  = Linear system size (100)
Np = Number of particles (100)
N  = Number of time steps (10000)
Nsample = Number of independent simulations (10)
```

2. Necessary data structures

```
/* lattice */
lattice = (int *) calloc((L*L),sizeof(int));

/* x and y coordinates of atoms */
xcoord = (int *) calloc(Np,sizeof(int));
ycoord = (int *) calloc(Np,sizeof(int));

/* help variables for calculating <r2> */
x0 = (int *) calloc(Np,sizeof(int));
y0 = (int *) calloc(Np,sizeof(int));
xt = (int *) calloc(Np,sizeof(int));
yt = (int *) calloc(Np,sizeof(int));
R2 = (double *) calloc(N,sizeof(double));
```

3. Initialization

```
/* Initialize all lattice sites to zero */
for(i=0; i<L*L; i++) lattice[i]=0;

/* Place particles randomly onto lattice */
for(i=0; i<Np; i++) {

  /* Choose a random site */
  x = L*ran_num(&seed);
  y = L*ran_num(&seed);

  /* Check if the site is already occupied */
  while (lattice[R(x,y)]==1) {
    /* If so, then choose a new site */
    x = L*ran_num(&seed);
    y = L*ran_num(&seed);
  }

  /* Mark the selected site occupied */
  lattice[R(x,y)]=1;

  /* Store x and y coordinates */
  x0[i]=xt[i]=xcoord[i]=x;
  y0[i]=yt[i]=ycoord[i]=y;
}
```

4. Choosing one atom randomly

```
/* random index */
atom = Np*ran_num(&seed);

/* x and y coordinates */
x = xp = xcoord[atom];
y = yp = ycoord[atom];
```

5. Choosing the jump direction randomly

```
/* Random direction */
dir = 4*ran_num(&seed);

/* Corresponding changes with pbc (moduloarithmetics would be
more effective, see e.g. Allen, Tildesley) */
switch(dir) {
case(0): /* left */
  x--; dx=-1; if(x<0) x=L-1;
  break;
case(1): /* right */
  x++; dx=1; if(x>=L) x=0;
  break;
case(2): /* down */
  y--; dy=-1; if(y<0) y=L-1;
  break;
case(3): /* up */
  y++; dy=1; if(y>=L) y=0;
  break;
}
```

6. Trying to perform the jump

```
/* Check if the chosen site is unoccupied */
if(lattice[R(x,y)]==0) {

  /* move the atom */
  lattice[R(xp,yp)]=0;
  lattice[R(x,y)]=1;

  /* new coordinates */
  xcoord[atom]=x;
  ycoord[atom]=y;

  /* relative coordinates */
  xt[atom]+=dx;
  yt[atom]+=dy;
  accept++;
}
```

7. Updating averages (after N_p random jump attempts)

```
for(i=0; i<Np; i++)
  R2[mcs] += (double)(xt[i]-x0[i])*(xt[i]-x0[i])
           + (double)(yt[i]-y0[i])*(yt[i]-y0[i]);
```

Results

Figure 3 shows the acceptance ratio of diffusion changes as a function of atom concentration on the surface. We notice that the acceptance ratio follows the relation $r_{acc} = 1 - \theta$. In other words, the acceptance ratio is equal to the fraction of unoccupied sites on the surface.

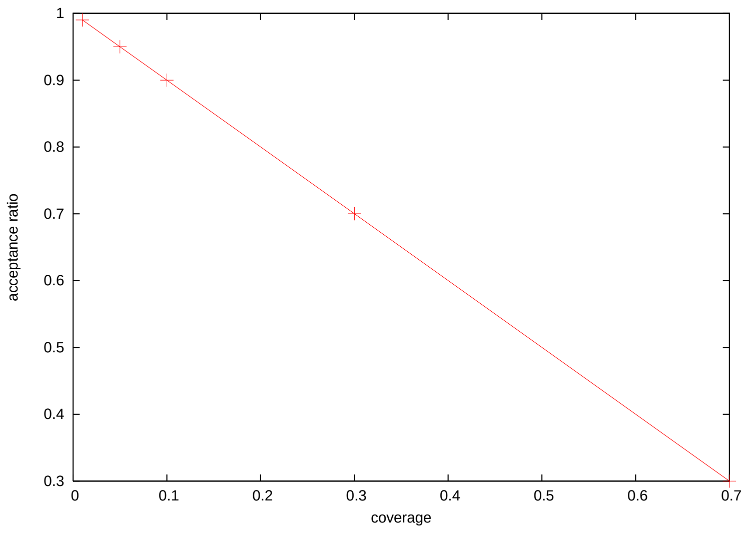

Figure 3: Acceptance ratio of diffusion jumps as a function of concentration.

Figure 4 shows the mean squared displacement of the atoms as a function of time t. The data are averages over 10 independent measurements (samples) and each independent value is an average over all the atoms in the system. For all concentrations, the data fall approximately on a straight line as expected for a group of random walkers.

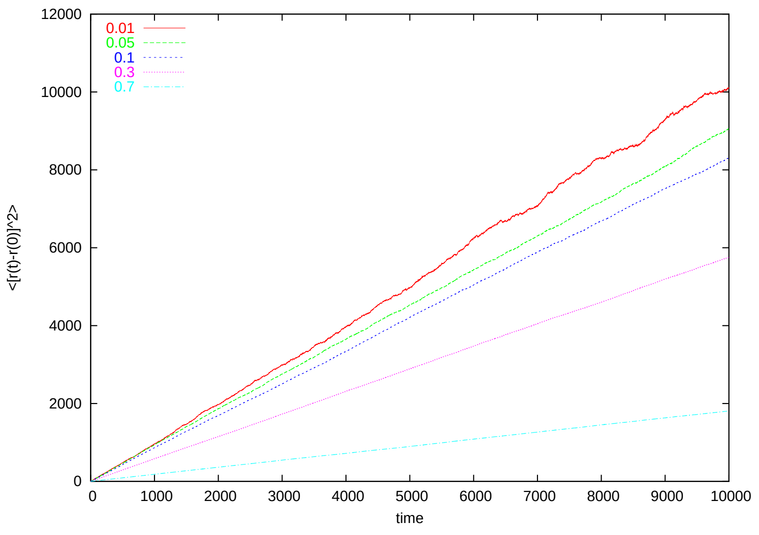

Figure 4: Mean squared displacement of atoms as a function of time for different concentrations.

1.4 Full diffusion model

The simple lattice gas model does not take into account the effects of temperature and mutual interactions between the diffusing atoms.

A refined approach, called the *full diffusion model*, is based on the so-called transition state theory in which the main idea is that a diffusion jump from site i to a neighbouring site j proceeds via a transition state. The two states are separated by a *potential energy barrier E_b*. If the atom is able to reach the top of the barrier, then the jump will proceed to site i with probability 1.

This leads to the following expression for the hopping rates (probability of a diffusion jump per unit time):

$$k(E_b, T) = k_0 \exp(-E_b/k_\mathrm{B}T)$$

where E_b is the energy barrier for hopping and $k_0 = 2k_\mathrm{B}T/h$ is an attempt frequency. The Boltzmann constant is $k_\mathrm{B} \approx 8.617385 \times 10^{-5}$ and the Planck's constant is $h \approx 4.1356692 \times 10^{-15}$.

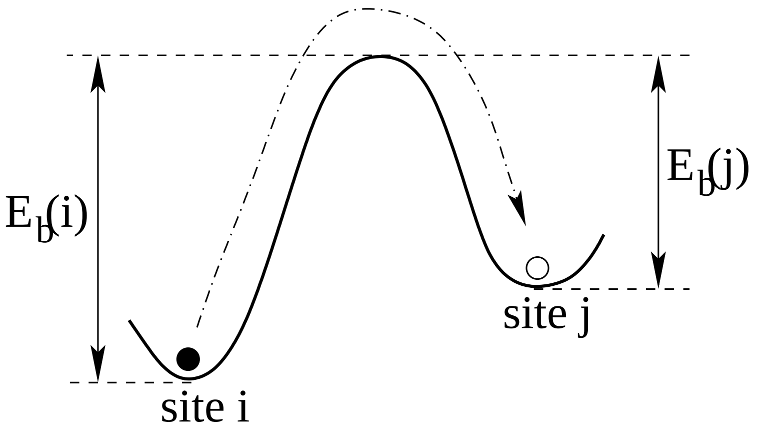

Figure 5: Diffusion barriers. The barrier for the jump from site i to j is E_i (depends only on site i). The barrier E_j for the reverse jump from j to i is in general different from E_i.

Nearest-neighbour interactions

In a basic diffusion model, the diffusing atoms interact with the substrate and with their nearest-neighbours. The diffusion barrier for a jump from site i to site j is given as a sum of two contributions: a site-independent surface energy term E_s and an in-plane bond-energy term E_n.

The total barrier to hopping is

$$E_b = E_s + nE_n$$

where n is the number of nearest neighbours of the atom at site i ($n = 0 - 4$).

Solid-on-solid model

A so-called solid-on-solid (SOS) approximation is used in most lattice-based growth simulations. This means that the surface consists of 'building blocks' which are piled on top of each other without leaving any vacancies or overhangs.

Thus in this model, each lattice site can be occupied by multiple atoms. This is interpreted as multiple atomic layers on top of the substrate. Implementing the SOS lattice is almost as simple as the lattice gas case: Each lattice site is denoted a number $n_i \geq 0$ such that n_i

gives the height of the atomic column at lattice site *i*.

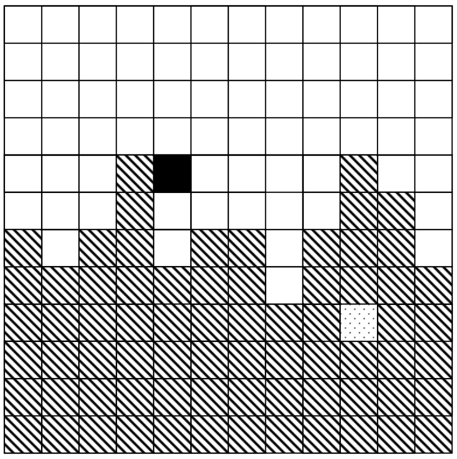

Figure 6: The solid-on-solid model. Vacancies (light gray square) or overhangs (black square) are not allowed.

1.5 Kinetic Monte Carlo or N-Fold Way

The Monte Carlo method was originally developed for calculation of average quantities in an *equilibrium* system. Nowadays, variants of the original approach have been developed for studies of various transport and growth phenomena. The ordinary MC (e.g. the Metropolis algorithm) cannot be used to study time-dependent properties because the sequence of generated states has no correspondence to the real dynamics of the system.

Kinetic Monte Carlo (KMC), also called the N-fold way, is a *non-equilibrium* method which has been designed to describe the *time evolution* of the system under study. In the KMC method, the sequence of generated states does follow the real path dictated by the dynamics of the system. Therefore, KMC can be used to model different diffusion processes and growth phenomena.

We will describe the KMC method here in connection with the solid-on-solid full-diffusion model. A growth process can be easily modelled by adding the possibility of deposition into the diffusion model.

The KMC algorithm is constructed by first identifying all possible *events* that may take place during the simulation. In our diffusion model, we have five different diffusion events defined by their rates. The hopping rate of each atom depends solely on its number of nearest neighbours. Thus all atoms which have an equal number of nearest neighbours diffuse with the same rate. This allows us to *group* the atoms into five different subsets according to the number of nearest neighbours.

Each step in the KMC algorithm consists of *one type of event*. The selected event is always realized (in the diffusion model, one atom always jumps). Note that this is different from the Metropolis algorithm where we always test for acceptance. Such testing is not done in the KMC method.

In practice, we do the following. Denote the total number of atoms by N. We have five different groups $\alpha = 0,1,2,3,4$ corresponding to the number of nearest neighbours n. We divide the atoms into these groups and construct five lists that contain the indices of the atoms belonging to each group. Let N_α denote the number of atoms that belong to group α.

The probability of a diffusion jump per unit time is given by $k_\alpha = k_0 \exp(-(E_s + \alpha E_n)/k_B T)$ for an atom in group α.

In order to choose an event, we calculate the *total rate* of all events:

$$R_{tot} = \sum_{\alpha=0}^{5} N_\alpha k_\alpha$$

Thus we multiply the individual rates by the number of atoms that diffuse with this rate. This gives us the probability of *any kind of diffusion jump* per unit time.

Once we know the total rate, we choose a random number $r \in [0, R_{tot}]$. The event corresponding to this random number is chosen (the figure below indicates how the selection is done). The events are indexed from 0 to 4. We begin by checking whether $r < N_0 k_0$. If this is true, then event 0 is selected. Else we check whether $r < N_0 k_0 + N_1 k_1$. If this is true, then event 1 is selected. We continue until we find the first event α for which $r < N_0 k_0 + N_1 k_1 + \ldots + N_\alpha k_\alpha$.

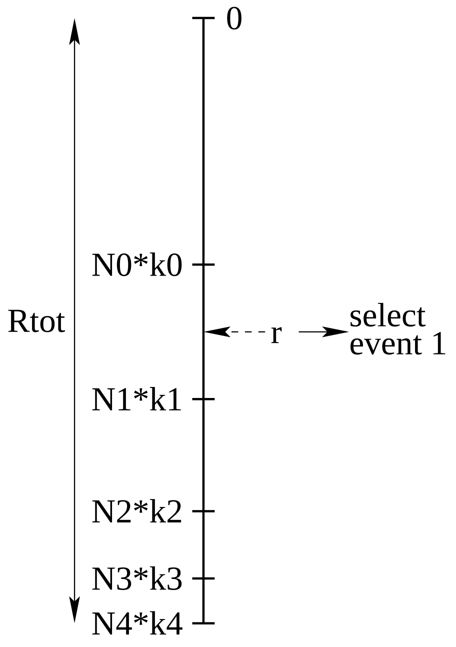

Figure 7: Selection of an event.

When we have chosen a given group (e.g. group 1 which means that an atom with one nearest neighbour is going to jump), we have to *choose one atom* from this group to perform the jump. This is again done randomly by choosing a random number $r \in [0, N_\alpha - 1]$.

Then finally, the *jump direction* is chosen randomly from the four possibilities (left, right, up or down). The selected atom is moved to the chosen site.

In this model, it is possible for the atom to jump *on top of another atom*. In that case, the height of the atomic column at the chosen site is simply incremented by 1 (e.g. $n_i = 2$ after the jump if the site was previously occupied by one atom. After such an event, we must remove the underlying atom from the list of moving atoms because it cannot diffuse while another atom is residing on top of it. If later the upper atom jumps away, then the underlying atom is reinserted into one of the diffusion groups depending on its number of nearest neighbours.

After a move, we must *update* the groups because the neighbours of the selected atom and its surrounding atoms have changed. The update is done by recalculating the number of nearest neighbours for the selected atom and for all atoms in the neighbouring sites of the initial and the final jump sites. We must also check initial and final sites for the possibility of having covered or revealed an atom as a consequence of the jump. It is important to consider only this small subset of atoms because recalculating the neighbours for the whole system after each jump would take much too long.

Some quantities (e.g. time, see below) are updated at this stage but the overall development of the system is usually monitored at some longer intervals. Remember that the system is not in equilibrium so we cannot calculate meaningful averages from a single simulation run. Instead, averages are calculated by performing several independent simulations.

Time in kinetic Monte Carlo

In equilibrium Monte Carlo simulations, we measure average quantities which do not change as a function of time and thus we need not know the real physical time. In these simulations, 'time' is usually measured in MC steps. The sequence of configurations has no correspondence to the real time evolution of the system which means that measuring real time is not possible even in principle.

In contrast, if we are simulating *dynamical* (time-dependent) processes, it is necessary to know the real time in order to calculate time-dependent physical quantities correctly. In some cases it is possible to find an easily measurable quantity which is proportional to real time. For example, in growth simulations where we have a constant flux of particles arriving on the surface and desorption is negligible, the number of atomic layers grown corresponds to real time.

Even if such direct proportionality cannot be found, we can still introduce the physical time into KMC simulations. Suppose that all the events are Poisson processes and no two events can take place simultaneously. The probability that at time t there were p events is given by

$$P(p) = \frac{(R_{tot}t)^p}{p!} e^{-R_{tot}t}$$

where R_{tot} is the total rate of all events.

The time interval between two successive events is a random variable τ with the distribution

$$P(\tau) = R_{tot} e^{-R_{tot}\tau}$$

The average waiting time is $\langle \tau \rangle = 1/R_{tot}$.

This allows us to generate a time increment between two successive events. After each event in the KMC algorithm, we draw another uniform random number $r \in [0,1]$. Before moving to the next event, the physical time is incremented by the amount

$$\Delta t = -\frac{1}{R_{tot}} \ln r$$

Note that the total rate changes after each movement, and thus the average of the time increment is not constant.

The *total physical time* is given by

$$t = \sum_{i=1}^{N_{MCS}} \Delta t_i$$

Algorithm - KMC simulation of surface diffusion

1. Distribute N particles initially to random lattice positions.
2. Group particles according to their number of nearest neighbours.
3. Calculate the total rate R_{tot}.
4. Choose a random number $r_1 \in [0, R_{tot}]$.
5. Select an event α which corresponds to r.
6. Select an atom randomly from group α by drawing a random number $r_2 \in [0, N_\alpha - 1]$.
7. Choose the direction of the diffusion jump randomly and perform the jump.
8. Update the division of atoms into groups.
9. Increment time and update quantities of interest.
10. Go to 3.

Measurement of averages during the simulation

If we are interested in measuring the average value of some quantity A during the simulation, we have to take into account that the time intervals differ from step to step.

In KMC simulations, average quantities are measured as *time averages* over N_{MCS} points in time:

$$\langle A \rangle = \frac{1}{t} \sum_{i=1}^{N_{MCS}} \Delta t_i A_i$$

where A_i is the value of A in configuration i and Δt_i is the time increment corresponding to the total rate of configuration i. N_{MCS} is the number of steps taken to measure the average.

This measurement is meaningful if we have reached some kind of a steady state in the simulation where the measured averages do not change as a function of time. Generally, it is more common to perform a number of simulations up to some time t and measure time-dependent average values $\langle A(t) \rangle$ by taking averages over the independent runs as a function of time.

Simulation of growth

It is now easy to extend the diffusion model into a full growth simulation. *Molecular beam epitaxy* (MBE) is a widely used technique for growing thin films on surfaces. In this method, new material is deposited by directing a beam of atoms or molecules onto the substrate surface. The atoms can diffuse on the surface but desorption is negligible under usual growth conditions.

The KMC method can be used to model MBE growth by adding *deposition* of new atoms into the list of possible events. This means that in addition to the five different diffusion events, atoms arrive randomly onto the lattice at a constant rate of F monolayers per second (ML/s). One monolayer is defined as one full layer of atoms which equals $L \times L$ atoms on the square lattice.

In the KMC algorithm, the total rate is now calculated as a sum of all the diffusion events and the deposition term:

$$R_{tot} = \sum_{\alpha=0}^{5} N_\alpha k_\alpha + L^2 F$$

An event is chosen at each step from the list of all possible events which now includes the deposition event. When deposition is selected, we randomly select one of the $L \times L$ lattice site and place a new atom there. This is done by simply adding 1 to the height of the atomic column at the chosen site.

After the deposition, we must add the new atom into the diffusion lists and update the neighbourhood of the deposition site. Then we can proceed to selecting another event.

In a growth simulation, the amount of deposited material gives us a direct way of measuring elapsed **time**. We can estimate the physical time by

$$t = \frac{N_p}{F}$$

where N_p is the total number of deposited particles and F is the deposition rate expressed in units of particles per second.

Note that in KMC simulations 'time' is always measured in the sense of an *average*. This is due to the fact that all the processes are random by nature and the rates are probabilities of something happening per unit time. However, if we measure the value of a given quantity A from several *independent* simulations at a given value of the time variable t, then the average of the measurements will give a correct estimate of the average value of A at time t.

Example - KMC simulation of MBE growth

As an application of the full-diffusion model, we discuss a case where this model is used to describe growth on metal surfaces. We do not go into details of particular materials but discuss some general effects.

The growth model is defined by setting the values of the parameters. A typical set of parameters for describing the growth of metals is given by: $E_s = 1.3$ eV, $F = 0.1$ ML/s, $T = 800$ K and $E_n = 0.3 - 1.0$ eV.

Keeping other parameter values fixed, we first examine the effect of changing the nearest-neighbour bonding energy E_n. The figure below shows snapshots obtained after deposition of 0.2 ML of new material for two different values of E_n.

The growth scenario can be described as follows. New atoms arrive at the surface at the rate F ML/s. The atoms diffuse on the surface until, by chance, they meet another diffusing atom. Now the rate for either atom to hop away is suppressed by the factor $\exp(-E_n/k_\mathrm{B}T)$. If any of the atoms gets another nearest neighbour, its diffusion rate is suppressed again by the factor $\exp(-E_n/k_\mathrm{B}T)$ and it becomes increasingly unlikely that the atom will detach from the island.

If $\exp(-E_n/k_\mathrm{B}T)$ is relatively large, it is easy for the atoms to break away if they have only one nearest neighbour. This is the case in Fig. 8(a), where most atoms have at least two neighbours and the islands have a *compact* shape. On the other hand, if $\exp(-E_n/k_\mathrm{B}T)$ is small, then atoms attach irreversibly to the first island they encounter. Fig. 8(b) shows an example of the resulting *fractal* islands.

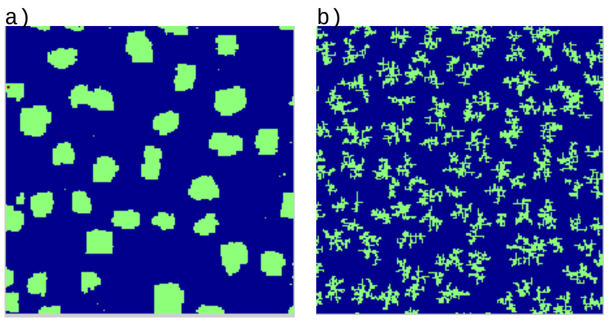

Figure 8: Islands grown on the surface of a substrate crystal. (a) Compact islands: $E_n = 0.3$ eV. (b) Fractal islands: $E_n = 1.0$ eV. The snapshots show 100×100 sections of the surface after deposition of 0.2 ML of new material.

Size distribution of islands

Looking at pictures of individual islands can give us a good hunch of what the behavior of the system is like, but a much more complete analysis is obtained by studying the *statistical properties* of a large collections of islands.

A typical quantity which characterizes the growth is the *distribution of island sizes*. In Figure 9, the size distribution has been calculated at a fixed temperature of 800 K after three different amounts of material were deposited on the surface. At very low coverages (0.05 ML of deposited material), the islands are all very small and the distribution is narrow and highly peaked. As more material is deposited onto the surface, the average island size increases and the distribution becomes wider.

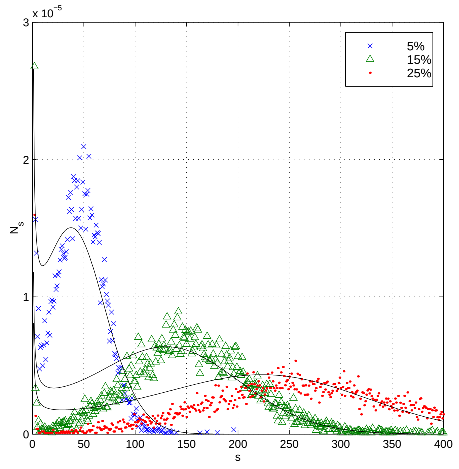

Figure 9: Size distribution of islands after deposition of 0.05 ML, 0.15 ML and 0.25 ML of new material. Temperature is 800 K. The islands have a compact shape ($E_n = 0.3$ eV). The solid lines are numerical solutions of analytic rate equations (differential equations).

The figure also shows solid lines which are obtained by solving a set of differential equations called the rate equations. These are based on an analytic theory of crystal growth. The agreement between the analytic theory and the KMC simulations is fairly good except that the rate equations overestimate the amount of very small islands. This is because the rate equations do not include any spatial information of the system and thus effects due to local correlations are lost.

(For a brief and clear description of N-fold way, see Landau, Binder: A Guide to Monte Carlo Simulations in Statistical Physics: 5.2.3 N-fold way and extensions.)