Friedrich-Alexander-Universität
Faculty of Engineering

Department Materials Science

WW8: Materials Simulation

Practical: Kinetic Monte Carlo Simulation

Leon Pyka (22030137)

Supervision: Dr. Frank Wendler

Contents

1 Introduction

The Monte Carlo method broadly refers to an extensive toolset of modeling methods that generate random numbers to solve problems. The underlying algorithm is the Metropolis algorithm. Unlike Molecular Dynamics, Monte Carlo methods do not follow the simulated particle trajectories but randomly perturb the system to assess the resulting system's properties compared to the preceding system. The *Kinetic Monte Carlo* (KMC) method incorporates kinetic features to the simulation. Each state transition is seen as a diffusive jump and can happen at time scales more extensive than in Molecular Dynamics (limited by the time-scale of atomic vibrations) [1].

1.1 General Algorithm of the rejection free KMC Method

We start with a system of possible states i and possible transitions $i \to j$. The transition rate is ν_{ij}. The following algorithm is reproduced from the problem statement [2].

1. start with state i at time t
2. evaluate total jump rate Γ
$$\Gamma = \sum_j \nu_{ij} \tag{1}$$
3. evaluate probability of jump by comparing ν_{ij} to the total jump rate (eq. 2a) and calculate the cumulative probability 2b
$$p_j = \frac{\nu_{ij}}{\Gamma} \tag{2a}$$
$$P_j = \sum_{j'<j} p_{ij'} \tag{2b}$$
4. generate a random number $R_1 \in [0, 1)$
5. chose state which satisfies $P_{ij} = \min P_{il}, P_{il} > R_1$ (where P_{il} is a list ordered in descending order, to make sure that high probability steps are more likely to be executed)
6. generate random number $R_2 \in [0, 1)$ and calculate jump time t_{ij} as
$$t_{ij} = -\frac{\ln(R_2)}{\Gamma} \tag{3}$$
7. update system configuration: $i \to j$ and update time $t = t + t_{ij}$
8. restart loop

2 Diffusion Problems Solved with KMC

2.1 Diffusion in a Cu thin film with time dependent diffusion rate

In this task, we're examining a sandwich structure consisting of a 100 nm thick Cu film and a Cu-10% Al film. Since Cu is a face-centered cubic (fcc) metal, we can simplify the setup to a 2D simple cubic lattice (only looking at the tilted {1 0 0}-plane). The diffusion coefficient is provided as follows:

$$D = 1.49 \cdot 10^{-7} \exp\Big(-\frac{136.1\text{kJ}}{\text{RT}}\Big)\frac{\text{m}^2}{\text{s}} \tag{4}$$

In 3D-the jump rate for e.g. Carbon at octahedral interstitial sites in α-iron is given as [3]:

$$\nu_{ij} = \frac{6D}{b^2} \tag{5}$$

so for a 2D-simple cubic with 4 possible moving directions it should be given as:

$$\nu_{ij} = \frac{6D}{b^2} \tag{6}$$

The jump rate ν_{ij} can be derived from D as follows:

$$\nu_{ij} = \frac{4D}{b^2} \tag{7}$$

with $b = 2.54$Å. As we are considering only one kind of diffusion process ν_{ij} is the same for all jumps so eq. 1 simplifies to:

$$\Gamma = n\nu_{ij} \tag{8}$$

and the probability from eq. 2a simplifies as stated below:

$$p_{ij} = \frac{\nu_{ij}}{\Gamma} = \frac{1}{n} \tag{9}$$

where n is the number of jumps. In the simulation the temperature needs to be increased linear to the time - from 293 K to 600 K. So the temperature at every time step (after incrementation according to the algorithm in section 1.1) is found as:

$$T_i = T_{\text{start}} + \frac{T_{\text{fin}} - T_{\text{start}}}{t_i / t_{\text{total}}} \tag{10}$$

where T_i is the current temperature at time-step t_i, T_{start} is the initial temperature, T_{fin} the temperature at the end and t_{total} the total time until the heating is finished. Running the simulation exposes some additional challenges. As the size of the time-step is inversely proportional to Γ. This means that the time-step at room temperature can become very long. It is so large that it can exceed the total runtime of the simulation in the first step. To prevent this, the initial rejection-free algorithm is updated to include the possibility of rejection. This is triggered when a certain time-step is exceeded as the time and temperature are updated in the range of the time-step but without performing a perturbation of the system. The second problem is the time complexity of the simulation. We must ensure that the system is updated after every move to prevent, e.g., Al atoms from switching sites with other Al-atoms. If this is checked for every jump, the time complexity instantly becomes of size $O(n^2)$. This can be tackled using a data frame storing the atom's coordinates and neighbors. In the initial implementation a `pandas.DataFrame` was used. The random moving direction was chosen from the neighbors list using the `pandas.sampel()` method. This resulted in a peculiar error displayed in Fig. 2.1. As all the atoms seem to have a preferred moving direction, this might expose a problem with the pseudo-randomness of the `pandas.sampel()` function. The tendency to choose the first moving direction (down) over the other direction shows that the function is not uniformly distributed for the required sample size.

Fig. 2.1 shows a successful implementation utilizing a `numpy.array` storing the individual atoms as objects carrying the type of the atom, the index of the neighboring atoms, the index of the sites with which the atom can switch places, and the number of switches the individual atom can perform. The flattening of the concentration gradient is qualitatively visible in fig. 3.

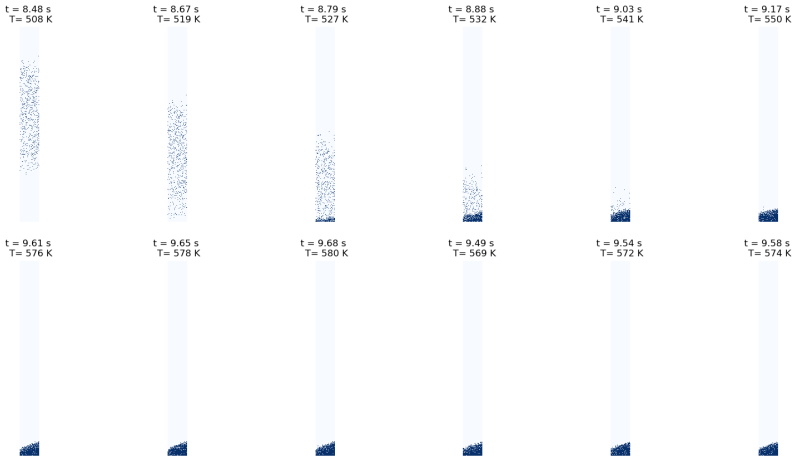

Figure 1: Failed implementation of diffusion in an Cu/Cu-10% Al-sandwich structure. The blue dots represent Al-atoms at substitutional sites.

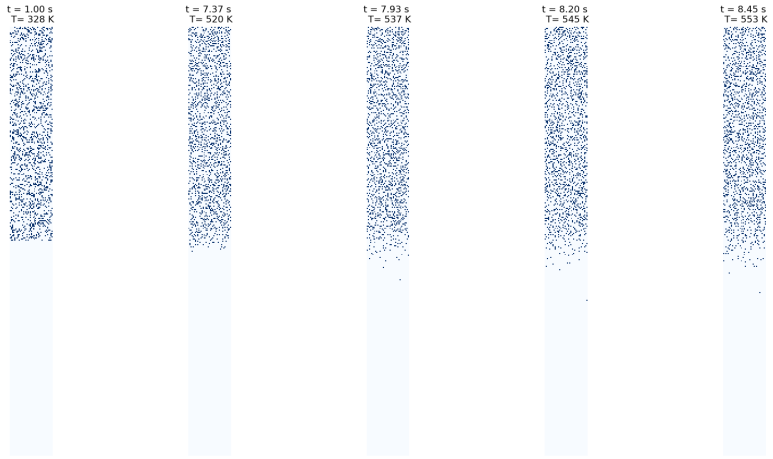

Figure 2: KMC-Simulation of a Cu/Cu-10% Al-sandwich structure. The blue dots represent Al-atoms at substitutional sites.

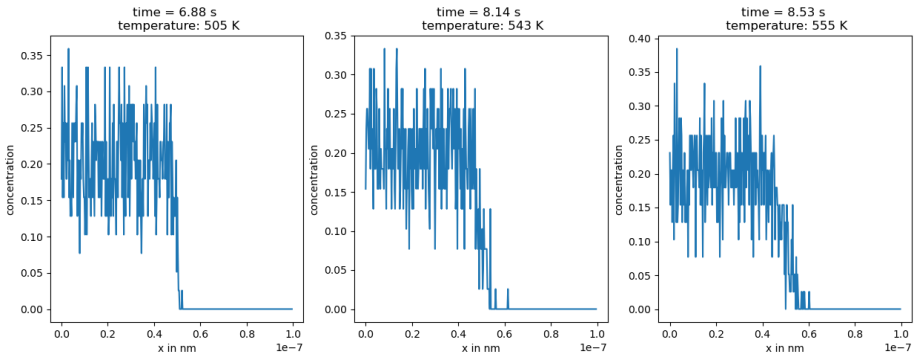

Figure 3: x-position against concentration of Al per row, evaluated different times and temperatures.

2.2 Coarse Grained Diffusion

In the second task, we consider the same set-up as in section 2.1 but with a layer thickness of 20 μm and a heating time of 10^6 s (≈10 days). As just increasing the simulation box size will make the method too time-intensive, we need to introduce a coarse-grained simulation. The following implementation aims to represent the film as a 1D array of N entries. Each entry carries the number of atoms in a specific "box" of Length L where if L_{total} = 20 nm is the sample thickness L is $L = L_{\text{total}}/N$. Running the simulation provides insights into a bug that distorts the results. This bug leads to the topmost entry in the array always becoming empty as the topmost entry is theoretically chosen equally often as the other entries - using pythons `random.choice` function. But as the following index can push al-atoms to the next and the previous index, whereas the first index can only move in one direction, this leads to a de-facto flow from $i = 0$ to $i = 1$. This can be omitted by including a (continue) (where the `random.choice` loop is executed, without decrementing the number of jumps) statement that is executed that is executed with equal probability to pushing from $i = 0$ to $i = 1$. The concentration is mapped in fig.4. The general gradient is similar to the atomic-scale simulation. In contrast, the extended resting period at the respective temperatures allows some Al-atoms to diffuse deeper into the pure Cu layer. The atomic scale simulation (see fig. 3) shows no trace of Al in the deeper layers. This supports the assumption that the time factor strongly influences the kinetics of the diffusion process.

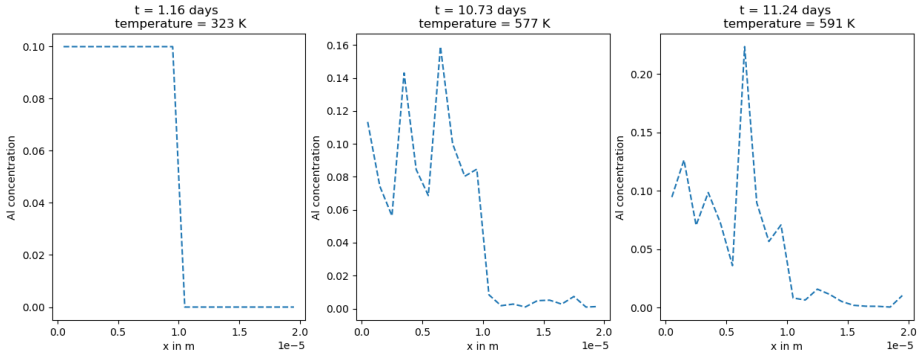

Figure 4: x-position against concentration of al per row at evaluated different times and temperatures in a coarse grained KMC-simulation of a Cu/Cu-10% Al-sandwich structure

References

1. Voter, A. F. *INTRODUCTION TO THE KINETIC MONTE CARLO METHOD* in *Radiation Effects in Solids* (eds Sickafus, K. E., Kotomin, E. A. & Uberuaga, B. P.) (Springer Netherlands, Dordrecht, 2007), 1–23. ISBN: 978-1-4020-5295-8.
2. Zaiser, M. *Kinetic Monte Carlo Simulation - Problem Statement*
3. Gottstein, G. in *Physical Foundations of Materials Science* (ed Gottstein, G.) 155–196 (Springer, Berlin, Heidelberg, 2004). ISBN: 978-3-662-09291-0. `https://doi.org/10.1007/978-3-662-09291-0_6` (2024).