Monte-Carlo simulation of two-dimensional grain growth

Author:
Christian Schmid

Supervisor:
Dr. Michael Leitner

Matrikelnummer: 0704670
Studienkennzahl: 033 621

November 25, 2011

Contents

1 Acknowledgements

I would like to thank Dr. Michael Leitner for his great supervision and for always being available for helpful discussion. My colleague Markus Macher answered many of my questions regarding the implementation of the simulation, and I also want to thank my parents for funding my studies.

2 Introduction

Many systems in nature have a cellular structure: A ‘big‘ object can be described as a network of smaller subdivisions – examples of this include fat eyes on a soup, soap bubbles and grains in a polycrystalline material. Although the latter will be used as a model example in the following simulation, the results can also be applied to different situations.
Most of these systems have in common that the total boundary between the cells is proportional to the energy of the systems, so a minimization of the energy corresponds to a minimization of the boundaries; the least energy is ‘lost‘ if the network consists only of one cell.
If the temperature of the system is positive, the minimization of the free energy, which is a diffusive process, can be simulated using a Monte-Carlo method: We introduce discrete time steps and decide stochastically at each step whether some proposed state is accepted as a new state of the system. The probability that this happens is given by a Boltzmann distribution.
We will specifically model a two-dimensional polycrystalline material; the atoms in its cells share a common lattice orientation.

3 Implementation

The model we used for this system is a two-dimensional plane graph. Its regions correspond to different grains, its edges to grain boundaries. To prevent surface effects, periodic boundary conditions were assumed, and to simplify calculations, all grains were supposed to have straight edges.

3.1 Voronoi diagrams

The initial data was constructed in the following way: N randomly distributed sites are chosen in a periodic region. Assume that grains are growing around each of these sites with the same speed, until they touch each other: The result will be a tessellation of the region, where a cell consists of those points that are closest to a fixed site – this is called a *Voronoi tessellation*. It is easy to convince oneself that the corresponding graph is (in the non-degenerate case) cubic; i.e., each vertex has exactly three neighbors. An example of such a Voronoi tessellation can be seen in Fig. 1.

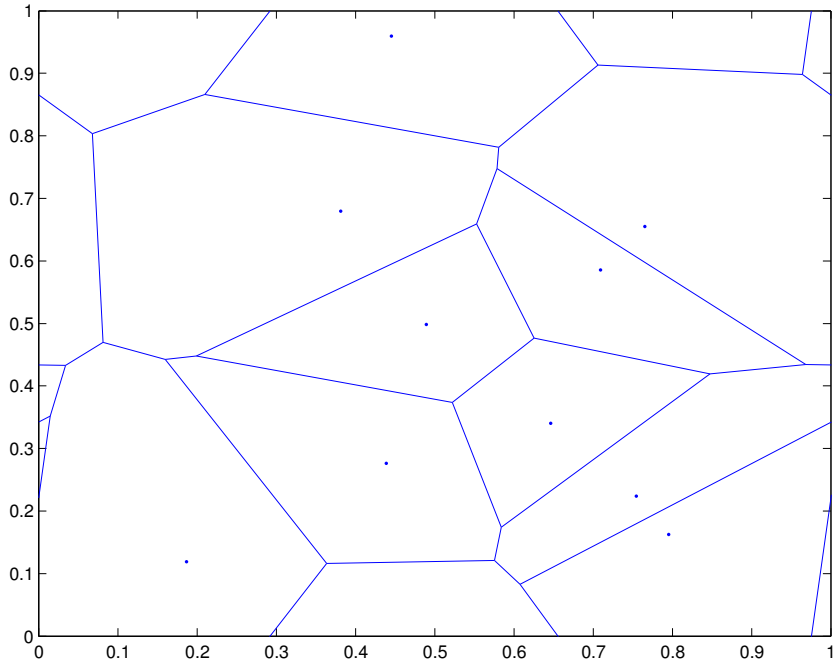

Figure 1: Periodic Voronoi tesselation generated by MATLAB

3.2 Metropolis algorithm

The actual simulation was implemented using a Metropolis algorithm: Assume that you're given a physical system with a set S of possible states, an energy function $E : S \to \mathbb{R}$ and an initial state $Y_0 = Y(t_0) \in S$ at time t_0. At each step of the algorithm, a new state Y' is sampled from a proposal distribution $f(Y'; Y_{n-1})$. Whether the perturbed state Y' is accepted as the new state Y_n depends on

$$\Delta E = E(Y') - E(Y_{n-1}),$$

the energy difference between these configurations.
If $\Delta E < 0$, the proposed state is accepted.
If $\Delta E > 0$, one draws a uniformly distributed random number $\xi \in [0, 1]$. One accepts the state if $\xi < e^{-\frac{\Delta E}{kT}}$, where T is the temperature of the system at time t_{n-1} and k is Boltzmann's constant. If $\xi > e^{-\frac{\Delta E}{kT}}$, one draws a new candidate for Y_n and repeats the test.

Thus, if we want to use this algorithm, we have to assume that the states of the system form a *Markov chain*; i.e., the probability that a specific state will be chosen is only dependent on the present state, not earlier ones.

3.3 Jumps

The energy function E in our case is the sum of all edge lengths in our graph; the value $\beta = \frac{1}{kT}$ is given by a constant factor. To generate a proposal state Y', we choose a random vertex and change its position by a small amount.

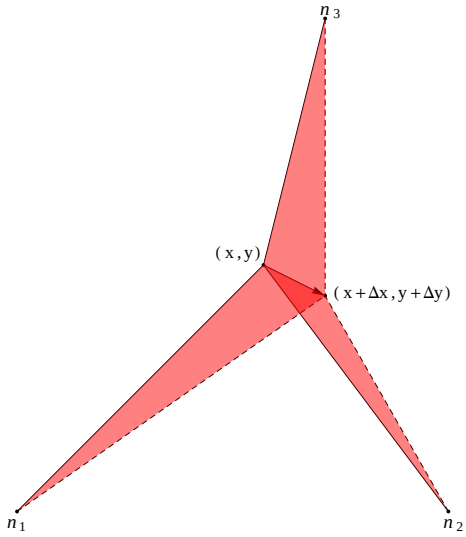

Figure 2: Change in grain areas after a jump from (x, y) to$(x + \Delta x, y + \Delta y)$

Because this step corresponds to a reorientation of atoms in a whole area (see Fig. 2), we impose a fixed maximal amount of this area, A_J, for a valid jump.

3.4 Topological changes

Physical grains or grain boundaries cannot be arbitrarily small, so there are some events where we have to check whether we need to change the topology of our graph: These are called T1 and T2 events (see Fig. 3).

In a T1 event, the distance between two vertices A and B is very small (see Fig. 4); we can see this as the situation where two grains, 1 and 2, 'collide' and separate the grains 3 and 4. Thus, we need to swap the edges such that grains 1 and 2 share a common boundary. We can do this in two ways: Either by interchanging A_1 with B_1 or A_2 with B_2 – for other combinations, the resulting graph would fail to be planar.
This step is done if the distance of A and B is smaller than a given, fixed distance, and the Boltzmann condition for the corresponding change in the energy is satisfied.
In a T2 event, the area of the triangle ABC (see Fig. 5) is so small that the corresponding physical grain vanishes. Thus, if it is smaller than a fixed minimal

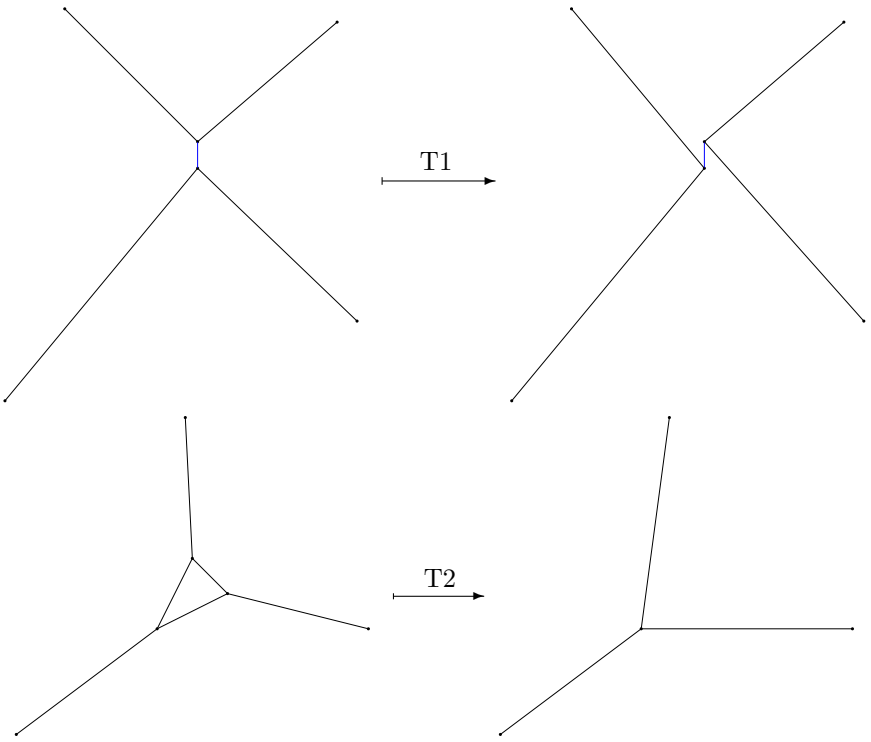

Figure 3: T1 and T2 events

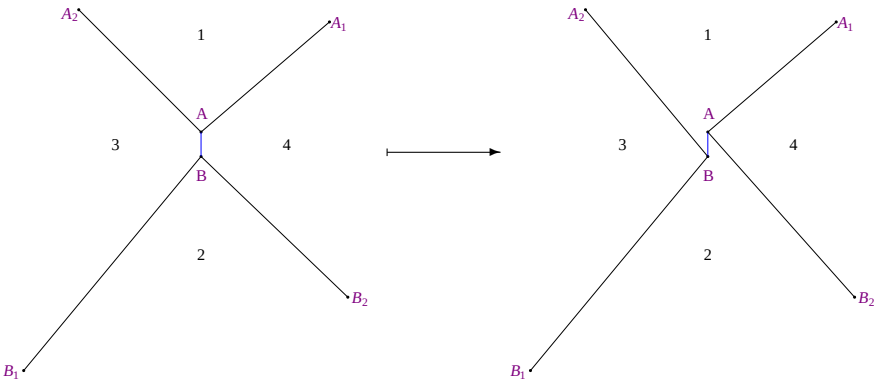

Figure 4: T1 event

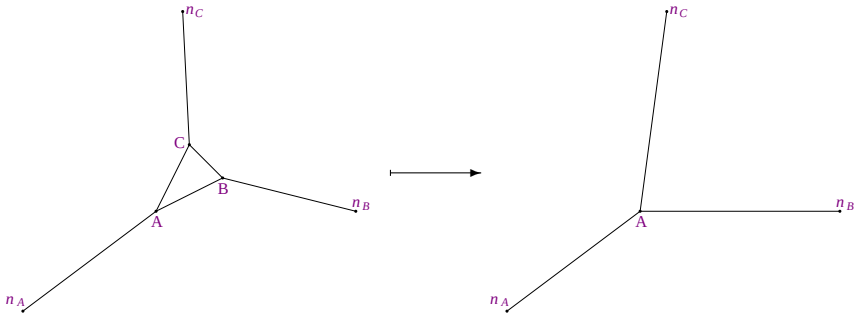

Figure 5: T2 event

area, A_{min}, the triangle is deleted and the vertex A inherits the neighbours n_B and n_C.

3.5 Problems

Of course, a naive implementation of this algorithm will lead to some problems that have to be fixed. First, we notice that the T1 event from Fig. 4 decreases the number of sides of the grains 3 and 4 by one. Thus, this step cannot be done if one of these polygons is a triangle. Our algorithm also breaks down in the case of a double triangle (see Fig. 6).

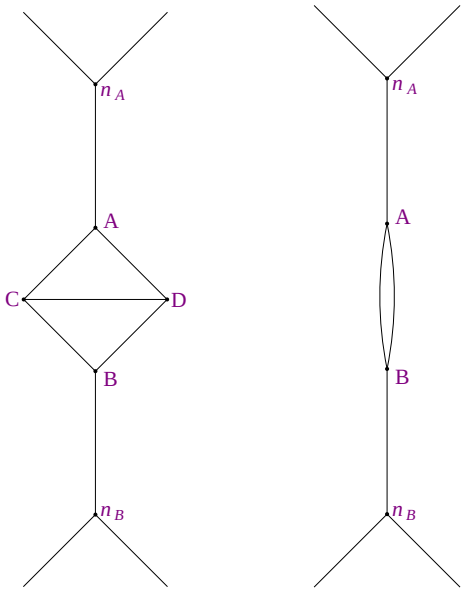

Figure 6: Double triangle before and after a T2 event

In this case, we directly connect vertex n_A and n_B, and delete both triangles. Note that this situation can (seldom) occur iteratively; in this case, we connect the two outermost vertices by a straight line.

Another problem is caused by the following effect: While the simulation is running, more and more grains are deleted. This implies that the mean edge length increases, but in our algorithm, the average length of one jump is constant, which means that the simulation 'slows down'. To prevent this, we rescale the following variables at each step i of the simulation:

$$A_J(\tau_i) = A_J(\tau_0)\frac{n(\tau_0)}{n(\tau_i)}, \tag{1}$$

$$A_{min}(\tau_i) = A_{min}(\tau_0)\frac{n(\tau_0)}{n(\tau_i)}, \tag{2}$$

$$\beta(\tau_i) = \beta(\tau_0)\sqrt{\frac{n(\tau_0)}{n(\tau_i)}}. \tag{3}$$

Here, τ_i is the 'simulation time' at which the i-th step is done, $A_J(\tau_i)$ is the maximal allowed jump area, $A_{min}(\tau_i)$ is the minimum area under which a triangle is deleted. The number $n(\tau_i)$ is the number of remaining vertices, and $\beta(\tau_i) = \frac{1}{kT(\tau_i)}$ is the thermodynamic factor occurring in the Boltzmann distribution.
A simple calculation shows that we can keep track of the 'real' time t by incrementing its value by $\left(\frac{1}{n}\right)^{3/2}$ at each step of the calculation.

4 Results and Conclusions

Taking these issues into account, the above algorithm was implemented in C with an initial number of $N := n(t_0) = 10^6$ nodes. The final runtime of the simulation was about 20 minutes on an Intel Core 2 Duo T9400 processor. In Fig. 7 you can see a section of the simulated graph for various values of remaining nodes n. Here, $\langle A \rangle$ is the average grain area (which changes during the simulation). Please note that this animation can only be viewed using a PDF viewer that supports JavaScript, e.g. Adobe Reader.
First, we want to find out how the introduced real time t depends on the remaining number of vertices n. We assume that

$$t(n) - t_0 = \int_N^n \frac{dt}{dn} = \alpha \int_N^n -\sqrt{\frac{1}{n^3}}dn = \alpha'(\frac{1}{\sqrt{n}} - \frac{1}{\sqrt{N}})$$

for some constant α'. If we set $t_0 = 1$, we expect $\alpha' = \sqrt{N}$. To check this, we make a log-log plot of t versus $\sqrt{\frac{N}{n}}$ (see Fig. 8).
We can see that the relationship is not fulfilled in the beginning of the simulation, but the behavior looks as expected afterwards.
Minimization of the total energy corresponds to a configuration with as few grains as possible. So we expect that, while some grains will grow bigger, most of them will shrink until they disappear. We want to find whether the growth rate correlates with the grain size. To measure this relationship, we let the simulation run until half of the nodes (or half of the grains) disappeared, so $n(t_1) = 500000$. At this time t_1, we measure the area of 1000 randomly

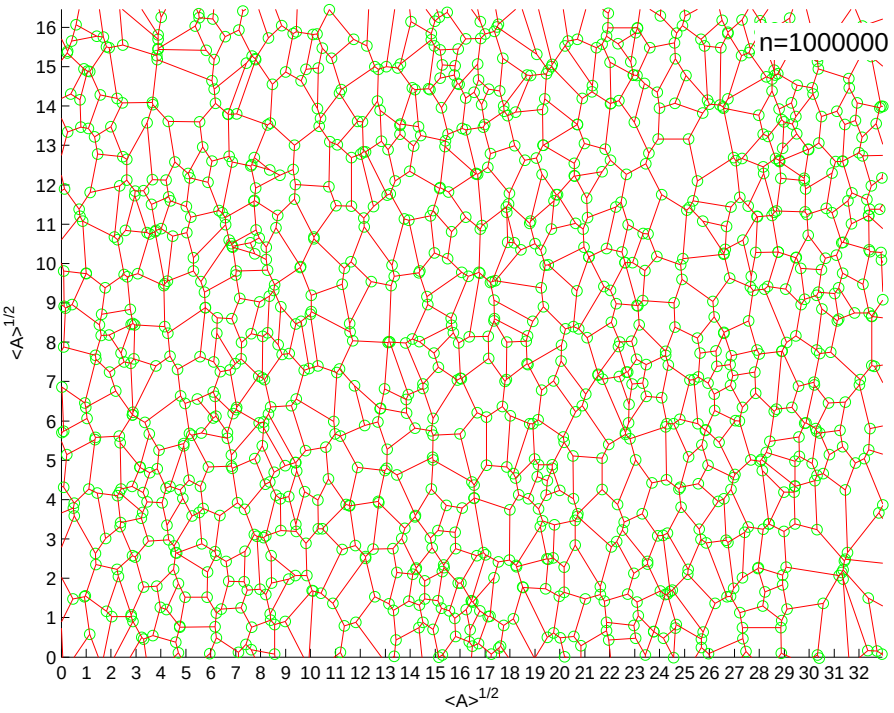

Figure 7: Animation of the simulation (use a JS enabled PDF viewer, click to start)

selected grains. We then continue with the simulation until we see an evolution caused by energy minimization, not just by a random walk. At this time t_2, each node jumped about 27 times and the remaining number of vertices was $n(t_2) = 480000$. We then plotted the area change $A(t_2) - A(t_1)$ against the initial area $A(t_1)$ (see Fig. 9). The values are scaled by $\langle A \rangle$, the average grain area at time t_1.

As expected, small grains tend to become smaller and disappear, bigger ones tend to increase their size.

We are also interested in the relative growth rate, which lets us determine which grains disappeared between t_1 and t_2. Thus, we plotted $R := \frac{A(t_2)-A(t_1)}{A(t_1)}$ against $A(t_1)$ (see Fig. 10).

Here, we can see that smaller grains shrink faster (relative to their size). This should not be surprising, as the distance of a jump is independent of the sizes of neighboring grains. As expected, all grains that vanished had a very small initial area.

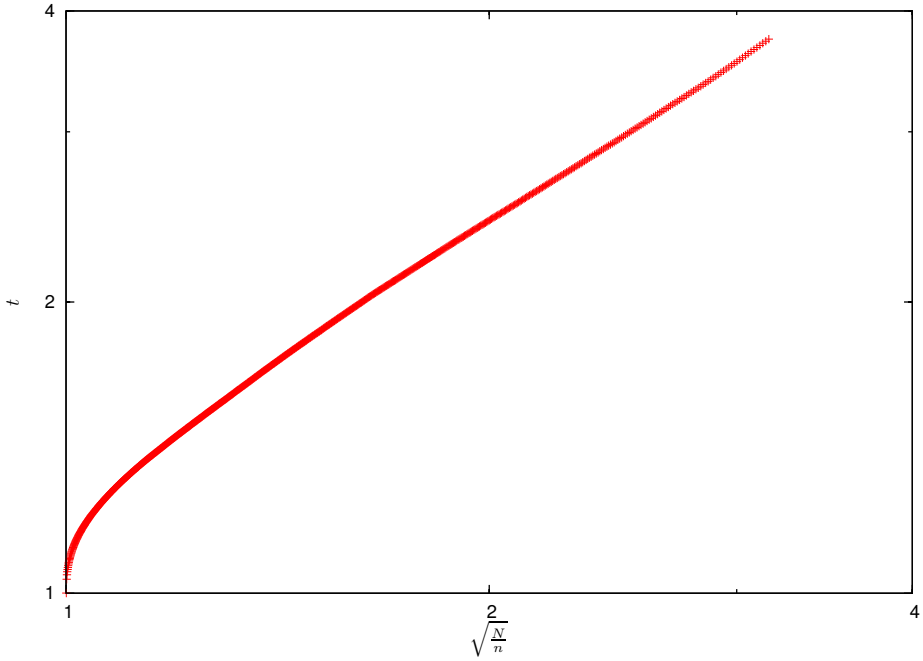

Figure 8: Log-log plot of the time t vs. $\frac{N}{n}$

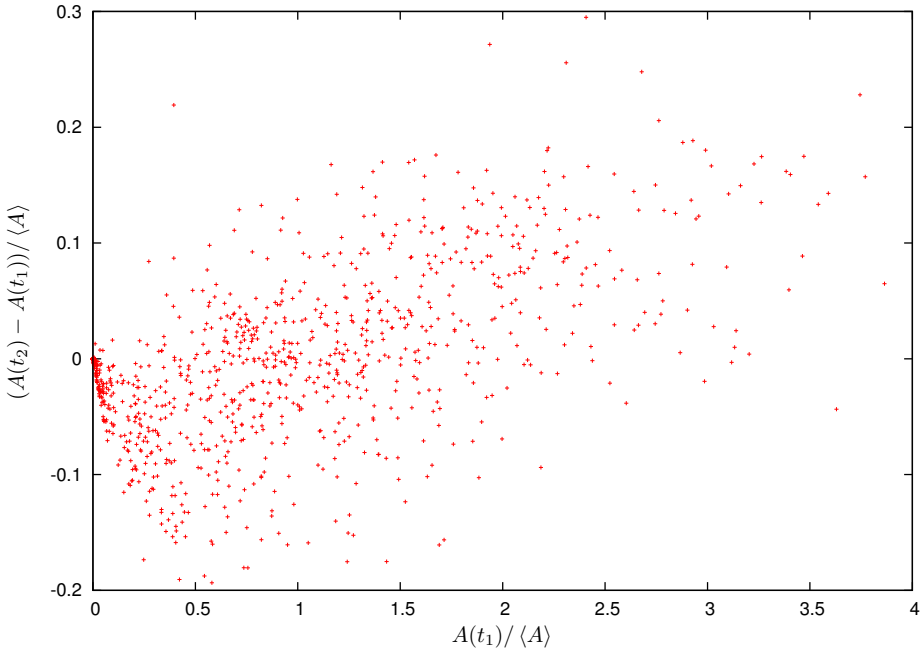

Figure 9: Absolute growth rate vs. size

Another interesting observable is the average distance between to neighboring nodes. We obviously expect it to grow as n decreases, so we made a log-log plot

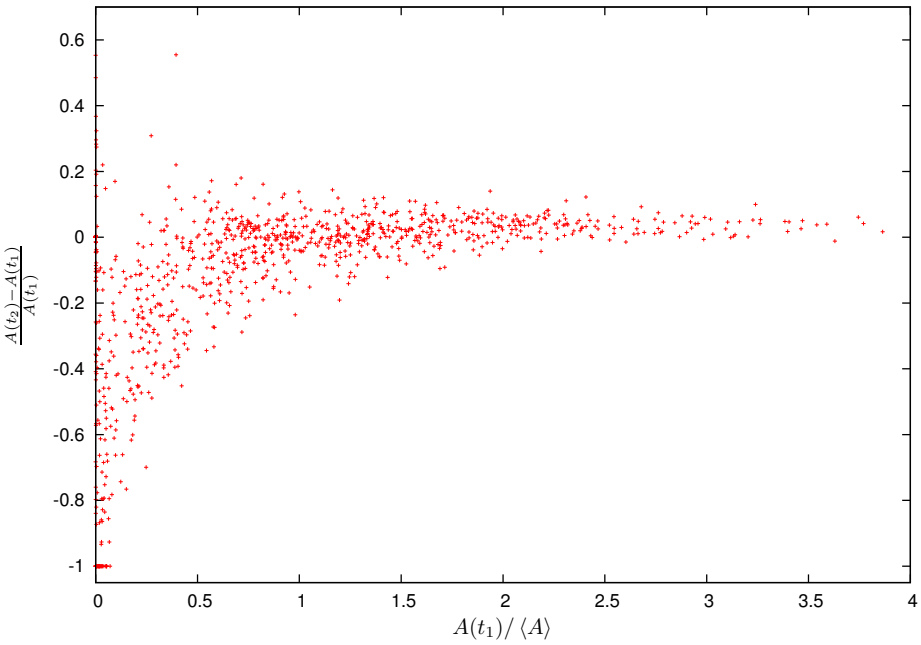

Figure 10: Relative growth rate vs. size

of these variables (see Fig. 11). Here, the scaling factor $\langle A\rangle$ is the average grain area at time t_0, with $n(t_0) = N$.

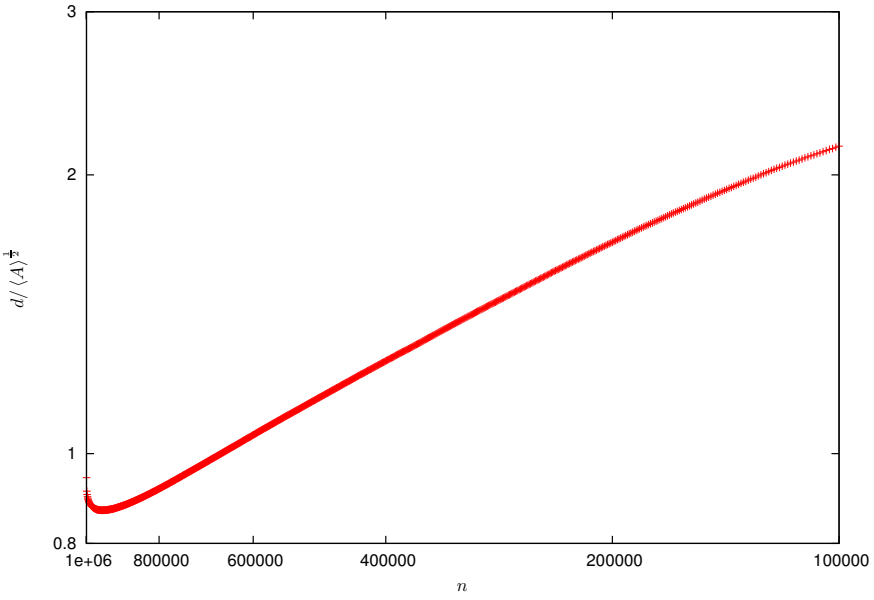

Figure 11: Log-log plot of the average distance d vs. the remaining number of nodes n

We see that the average distance decreases in the beginning, but soon grows

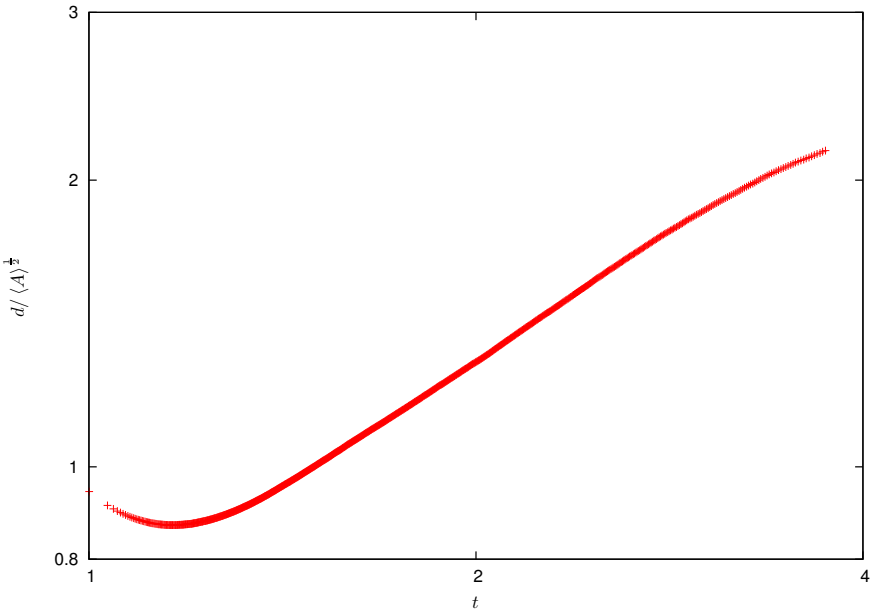

Figure 12: Log-log plot of the average distance d vs. time t

again – but as the log-log plot is non-linear, we cannot find an obvious relationship between these variables. It might be more interesting to plot the average distance against t (see Fig. 12). This is still not linear, but in the log-log plot, the part in the middle has a slope of about 0.54, which suggest a square-root relationship.

The average number of sides of the polygons is always 6. In Fig. 13 you can see with which probability a randomly chosen polygon has a given number of sides. We notice that the distributions are very similar for different values of n, until the end, where the number of quadrilaterals increases significantly, and grains with a higher number of edges become common.
Because we delete only triangles, we assume that polygons with a smaller number of sides have a smaller area. We can clearly see this in Fig. 14, where the values for the areas are scaled by $\langle A\rangle$, the average grain size for n remaining vertices. We also notice that triangles in the original Voronoi tessellation are bigger, but shrink very fast.

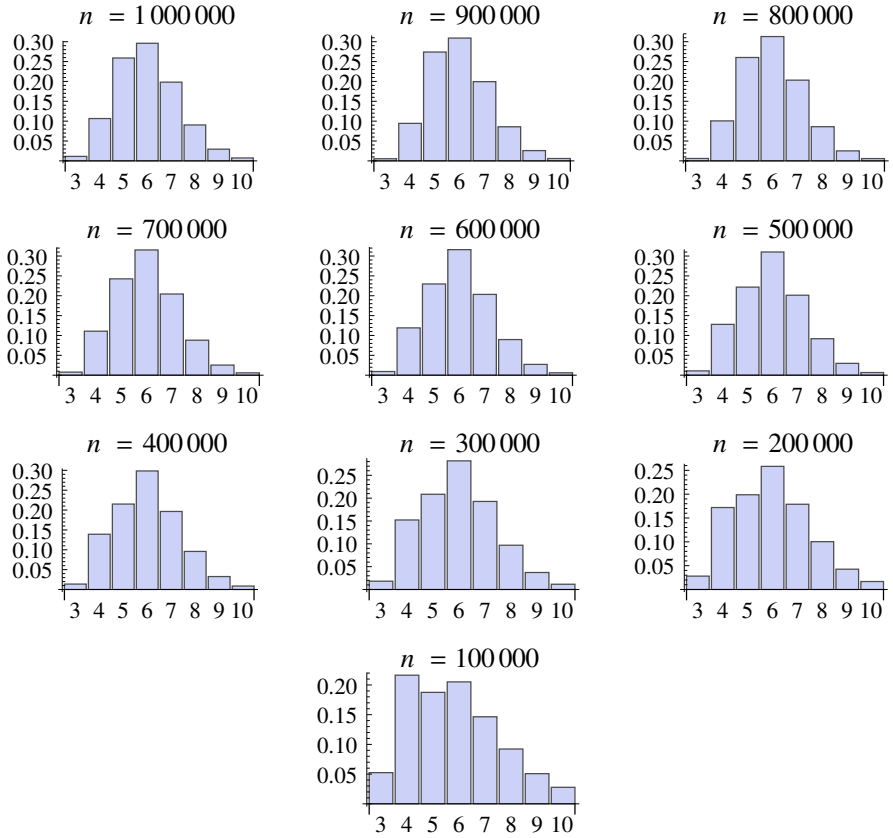

Figure 13: Grain side number frequency for different values of n.

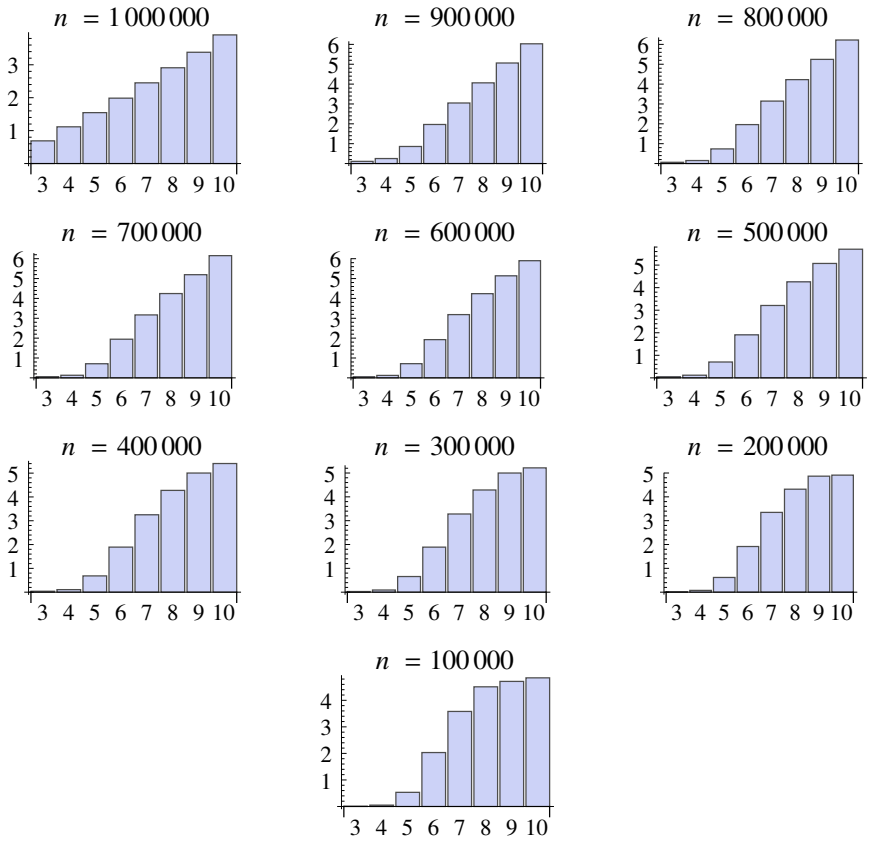

Figure 14: Average size for polygons vs. their size for different values of n.

The theoretical relationship between a polygon's growth rate and the number of its sides is given by the *Neumann-Mullins equation* (cf. [1]):

$$\frac{dA_n}{dt} = K(n-6), \tag{4}$$

where K is a constant and A_n the area of a polygon with n sides. It has already been noted in [2] that this relation is not observed in simulations with straight edges. In Fig. 15, where $\langle A \rangle$ is again the average grain size at time t_0, you can see that it is not observed for the growth rate of $\langle A \rangle$ (but this cannot be deduced from the Neumann-Mullins equation anyway): For example, Eq. (4) predicts that a grain with 5 sides should shrink, but this only happens in the beginning; then, at least their average size is increasing. The average sizes are also plotted w.r.t. $\frac{1}{n}$ in Fig. 16, which shows an approximately linear relationship.

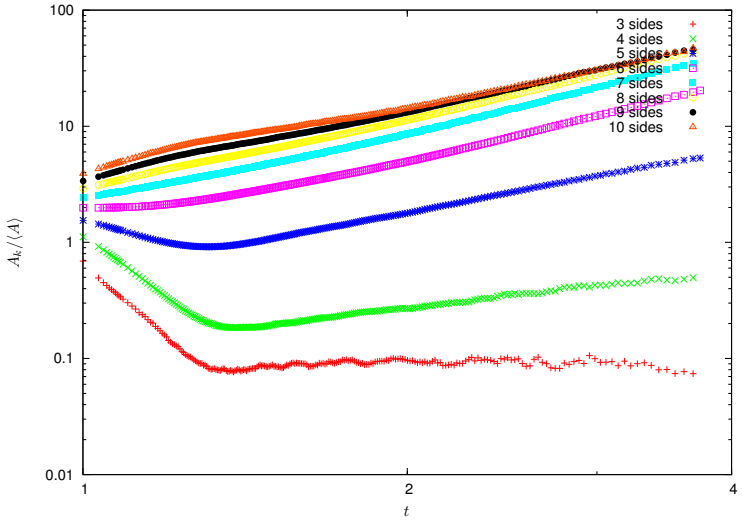

Figure 15: Log-log plot of the average size A_k of a k-gon scaled by $\langle A\rangle$, the average size of a grain at time t_0, vs. t.

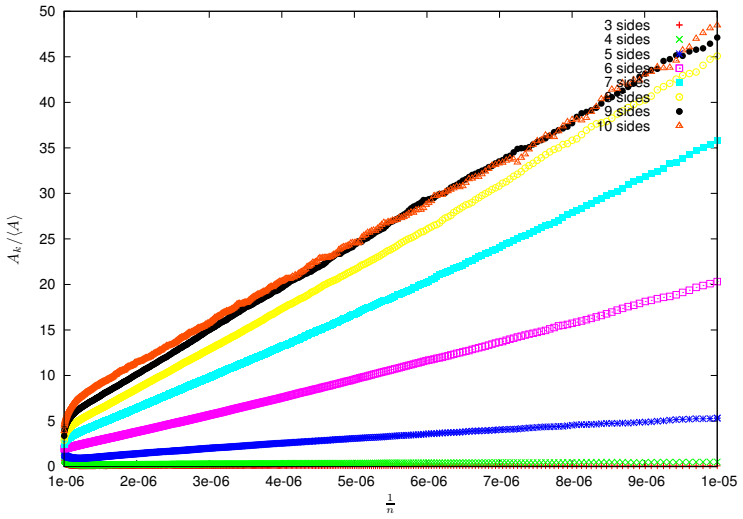

Figure 16: Average size A_k of a k-gon scaled by $\langle A\rangle$, the average size of a grain at time t_0, vs. $\frac{1}{n}$.

To see whether the predictions of Eq. (4) hold for individual grains, we can look at the distributions of the growth rate of all polygons for each number of sides separately; this was done in Fig. 17 and combines data of 10 runs of the simulation with 10^6 initial grains each, the growth rate was calculated as in Fig. 10.
Here we see that most grains with less than six sides have a decreasing area, while those with more than six sides mostly increase in size.
To see how many grains of different sizes there are, we plotted the area distribution in Fig. 18. The graph for polygons with five sides or less does not have a local maximum. It seems that most of the grains would have already disappeared in nature, but still need to undergo some topological transformation until they are removed in the simulation.

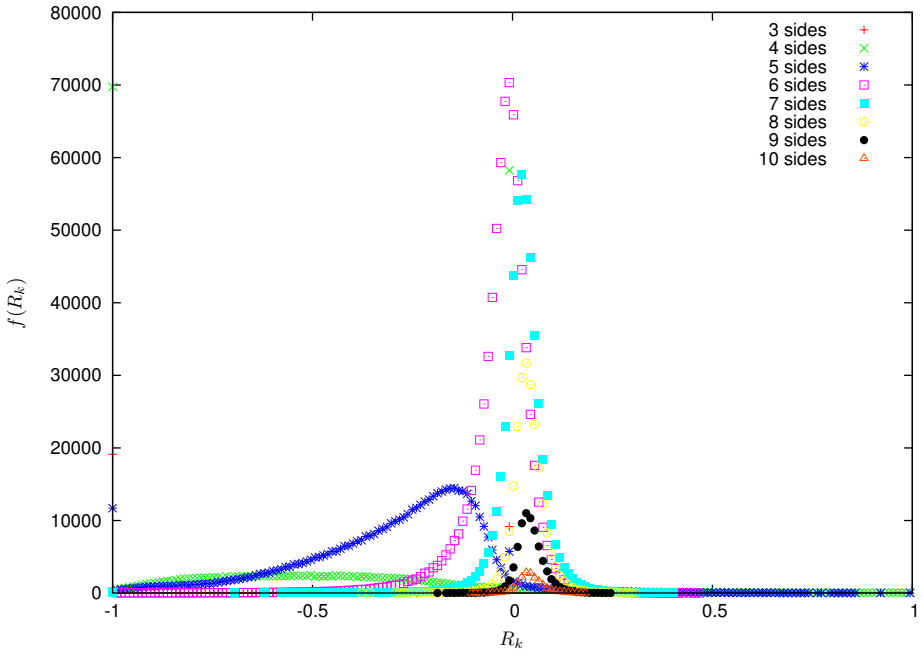

Figure 17: Frequency $f(R_k)$ of the relative growth rate $R_k = \frac{A_k(t_2)-A_k(t_1)}{A_k(t_1)}$ of k-gons

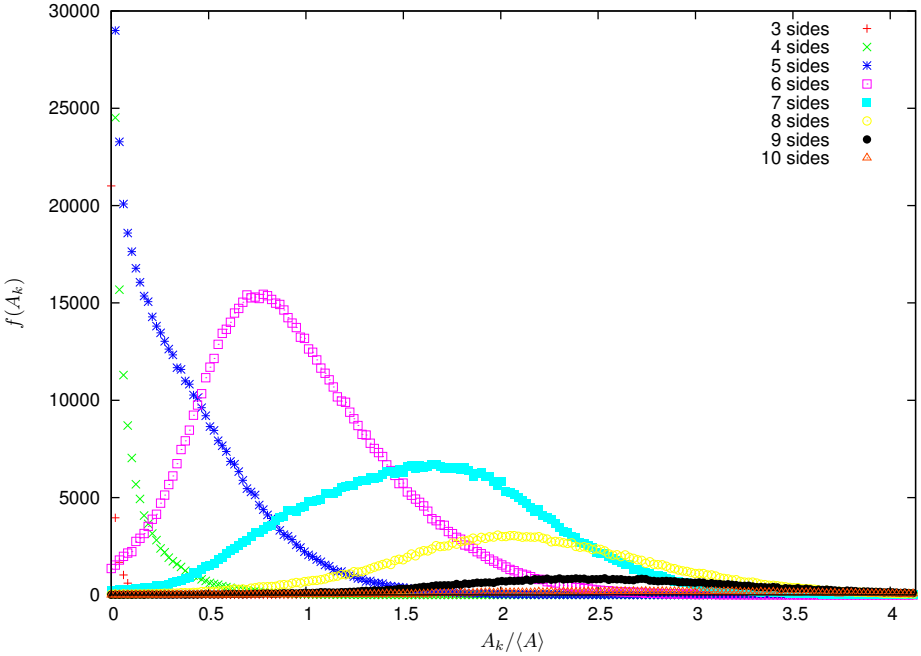

Figure 18: Frequency $f(A_k)$ of k-gon areas scaled by average area $\langle A\rangle$.

5 Outlook

The most obvious shortcoming of the model is that it doesn't allow curved boundaries. These could be approximated by introducing some additional points on edges and using them, for example, as B-spline control points. The problem of the disproportionate amount of quadrilaterals and small polygons with five sides at the end of the simulation could be caused by complicated topological situations; the author thinks that a better algorithm for dealing with double triangles could resolve this.

References

[1] W. W. Mullins. Two-Dimensional Motion of Idealized Grain Boundaries. *Journal of Applied Physics*, 27:900–904, August 1956.

[2] J.A. Glazier and D. Weaire. The kinetics of cellular patterns. *Journal of Physics: Condensed Matter*, 4:1867, 1992.