NTNU – Trondheim
Norwegian University of
Science and Technology

DLA based agglomeration of aerosols

Simen Berntsen
November 23, 2015

Specialization project applied physics
Department of Physics
Norwegian University of Science and Technology (NTNU).

Supervisor: Alex Hansen.

Contents

1 Introduction

This project has been on taking a Diffusion-Limited Aggregation (DLA) approach to the agglomeration of aerosols in the lower atmosphere in relation to emission from fossil fuelled based combustion (etc.) causing emission of dry carbon. Diffusion-Limited Aggregation is an algorithmic approach to a growth process where limited supply of mass is the limiting factor. A number of different growth models have been proposed, however the original one was proposed by T. A. Witten Jr. and L. M. Sanders [1]. This model builds on the Eden model, first proposed by M. Eden [2], but with a random walker determining the aggregation. A more dynamic model for agglomeration of moving clusters was proposed by P. Meakin [3]. This model was analysed using the scaling relation suggested by T. Vicsek and F. Family [4]. The goal of this project has been to use these methods, particularly the diffusion limited cluster aggregation (DLCA), to model the agglomeration of carbon immediately after emission, and to compare the results to what is experimentally observed by Scarnato et al. [5]. This should be more tied together! Make the point that it has not been to include the mineral dust into the model, but to recreate the carbon clusters. Should also be revised when all the "work" is done, to see what was accomplished, and not promise things you don't deliver.

2 Theory

2.1 Fractal dimension

The clusters are analysed according to their fractal dimension (Fraunhofer dimension), defined by their radius of gyration according to the following formula.

$$N = k_0 \left(\frac{R_g}{a} \right)^{d_f}, \tag{1}$$

where N is the number of particles in the cluster, k_0 is the fractal prefactor (also called structural coefficient), a is the diameter of the particles, and d_f is the fractal dimension. The radius of gyration, R_g, is defined as

$$R_g = \frac{1}{N} \sum_{i=1}^{N} r_i, \tag{2}$$

where r_i is the distance from particle number i to the centre of mass R_{CM}. Mention the substitution $M \leftrightarrow N$ because of $m_i = 1$.

2.2 Family-Vicsek scaling

A scaling relation for the cluster size distribution $n_s(t)$ has been found by F. Family and T. Vicsek [4]. ??? Write about scaling relation, and how this is used to analyze moving clusters. The distribution is defined as $n_s(t) = N_s(t)/L^2$, where $N_s(t)$ is the number of clusters containing s particles after time t. L is the lattice size (???L^2?). It has been found that $n_s(t)$ scales as a power-law

$$n_s(t) \sim t^{-w} s^{-\tau} f\left(\frac{s}{t^z} \right), \tag{3}$$

where w, τ and z are fitting (??? not sure if correct word) parameters. One should point out the importance of how time and step length scales in the

simulation, as this is a numerical experiment. This is due to diffusion and random walks being related through

$$D = \frac{(\Delta x)^2}{2\Delta t}. \tag{4}$$

Here D is the diffusion constant, Δx is the step size og the random walker, and Δt is the time step.

3 Method

In order to be able to design an algorithm for diffusion limited cluster aggregation (DLCA), some familiarity to the field was required. As an introduction and preparation, the case of regulare DLA was investigated, both on and off lattice. The main interest of section 3.1 is just to prepare for 3.2, where a more general DLCA algorithm has been developed. However, a lot of interesting aspects that are usefull to the DLCA algorithm are uncovered.

3.1 Diffusion limited aggregation

The fundamental algorithm of DLA, as described by Witten and Sanders [1] is as follows; a seed particle (the particle we follow) is placed at the centre. This particle will be the basis of the growing cluster. Another particle is then introduced at some distance from this seeding particle. The starting position of this particle can in principle be infinitely far away from the particle, but for practical matters, it is often placed close to the seeding particle. This particle will perform a random walk in the space around the seeding particle, with the intention that at some point in time, the walking particle will hit (i.e. be sufficiently close to) the seeding particle, and the two will clump together.

All of the following algorithms have been implemented using C++ 14. ??? Don't know about the following part, but: The code was run on OS X Yosemite 2,7 GHz Intel Core i5 8 GB 1867 MHz DDR3 system.

3.1.1 On-lattice DLA

For the on-lattice case, the algortihm is pretty straight forward, as is described in the beginning of section 3. Thus it will only briefly be discussed here, with the main focus being on the off-lattice version. Special for the on-lattice case is that the particle can only move in discrete steps and directions. That is, it may only move along the axis (either north, south, east or west), but not diagonally. More specific for the simulation run in this project is that the walking particles would always start at a random (uniformly distributed) point on the perimeter of the grid. If it was to take a step causing it to move outside of the grid, it would restart somewhere else on the perimeter. Although this cannot be done without some loss of generality, it is still done to increase the simulation speed. ??? should look into this more?

Since the on-lattice simulation is of less interest compared to the off-lattice case, only a crude on-lattice simulation was performed. The results of the on-lattice simulations can be found in ??? Link to section with results for on-lattice DLA simulations.

3.1.2 Off-lattice DLA

In off-lattice DLA simulations, the particle is no longer restricted to move along any axis. This means it can move freely in an approximately continous space, and any direction. This necessitates a more complex algorithm than for the on-lattice case. The algorithm used for the off-lattice DLA simulation is based on the one suggested by Kuijpers et. al. [6]. In this algorithm, one uses an on-lattice approximation for storing the positions of the particles in the cluster, thus being more RAM dependant than other algorithms, but with the benefit of reducing the calculation of sites viable for hits. Like in section 3.1.1, the particle starts along a circle of radius R_{start} from the cluster, and will be killed if the wander too far from the cluster. This limit is set to $R_{kill} = 5R_{cluster}$, where $R_{cluster}$ is the radius of the smallest circle spanning the cluster.

??? Write about the hit criteria! This is also used for the off-lattice DLCA program. As one goes from discrete to continous space, the algorithm must calculate if a hit is taking place. It is no longer enough to check if the neighbouring cell is occipied, so this is done according to figure 1. Cartesian coordinates were used, and the exact coordinates og particles eligible for hits are extracted (see section ??? LINK TO APPROPRIATE SECTION HERE!. As all particles are defined from their center, one has to take into account their extent (???). The distances a and b are found from geometric inspection of figure 1, giving

$$a = x_p - (x_0 + L_{hit}\cos(\alpha)) \tag{5}$$
$$b = y_p - (y_0 + L_{hit}\sin(\alpha)). \tag{6}$$

Inserting (5) and (6) into $d_p^2 = a^2 + b^2$ and rearranging gives

$$\begin{aligned} d_p^2 =& L_{hit}^2 + 2\left[(x_0 - x_p)\cos(\alpha) + (y_0 - y_p)\sin(\alpha)\right] L_{hit} \\ &+ \left[x_0^2 + x_p^2 + y_0^2 + y_p^2 - 2(x_0x_p + y_0y_p)\right]. \end{aligned} \tag{7}$$

This can be written on the form

$$AL_{hit}^2 + BL_{hit} + C = 0, \tag{8}$$

with parameters defined by

$$A = 1 \tag{9}$$
$$B = 2\left[(x_0 - x_p)\cos(\alpha) + (y_0 - y_p)\sin(\alpha)\right] \tag{10}$$
$$C = (x_p - x_0)^2 + (y_p - y_0)^2 - d_p^2. \tag{11}$$

Equation (8) may be solved as a usual polynomial of second orders, which means it may or may not have a solution. The algortihm is thus designed to take all possible situations into account. ??? May write all five cases and how the code treats them, or may just link to the Kuijpers paper?

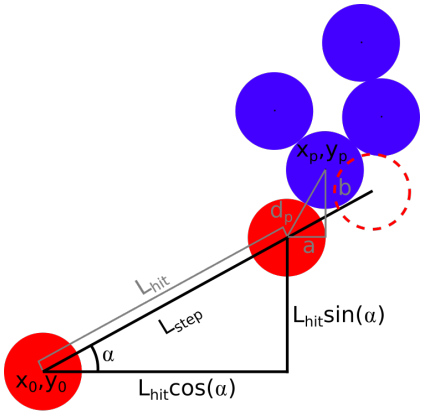
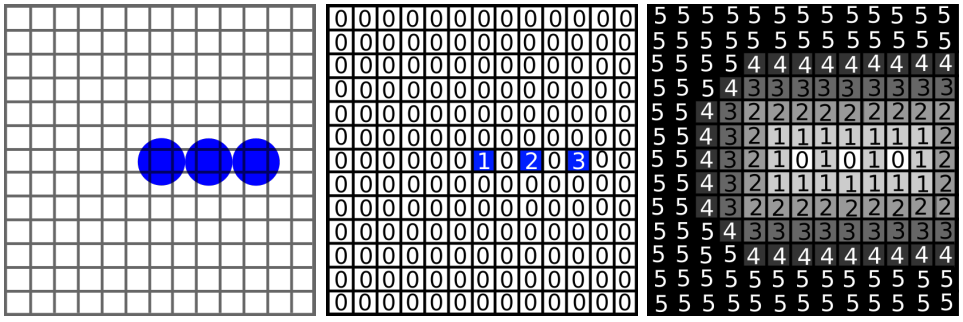


Figure 1: Schematic of the hit calculation for off-lattice simulations. Here, the red circles are the positions of the walking particle, and the blue are particles already in the cluster. The walking particle is launched from position x_0, y_0 in the α direction with the horizontal plane. The dotted red line indicates where the walking particle would end up, if it was to take a full step of length L_{step}. As the walking particle will collide with the particle located in x_p, y_p, it cannot do the full step, but rather a smaller of length L_{hit}. The distance d_p is the diameter of the particles, while a and b are geometrical distances.

With the new hit calculation routine, the concept of this new algorithm is as follows: The program uses three arrays (illustrated in figure 2), which together keeps track of all information of interest. The first one, $\mathbf{A}$, is an $N \times 2$ array (N is number of particles), in which the exact coordinates of the particles are stored. These coordinates are then mapped to integer values, so that they may correspond to a cell number in $\mathbf{B}$, an $n \times n$ (n is the size of the grid used for approximations) array in which all the elements are either 0 or the label of the particle. See figure 2a and 2b for an illustration of

this. In this context, label refers to the number given to the particle when it attaches to the cluster. Starting from the original seed particle, which is given label 1, the particles are labeled in ascending order, meaning the first particle to hit the seed particle will get label 2 and so on. The final array, $\mathbf{C}$ (also of dimension $n \times n$), stores the distance from each cell to the closest particle, as shown in figure 2c. This is done in order to have quick access to the distance from a walking particle to a point on the cluster. The great benefit of this algorithm lies in being able to take steps of varying length, depending on the distance from the cluster, which is only possible without loss of generality if the particle is far away from the cluster. There is a set a maximum distance D_{max} for the distances in $\mathbf{C}$, in order to reduce the amount of calculations required per time iteration. D_{max} balances the length of the steps the particle may take (longer steps results in a faster program), to the number of particles needed to be checked (which slows the program down).



(a) Figure of the example configuration.

(b) How the labels of the particles would be. The original seeding particle would be numbered 1, and the other particles are labeled as they hit the cluster.

(c) Illustration of the array with the rounded off distances. Knowing what cell the walker is in, it is easy to determine its distance to the cluster.

Figure 2: Illustration of a dummy configuration of particles, and how each array work. The seed particle is in all cases placed in a 13×13 array, at position (7,7). For simplicity, the other particles are not random walkers in this dummy case, but move in from right to left, along the horizontal axis.

Throughout the process, the walking particle will take steps depending on its distance from the cluster. The following three criterias determines the different regimes the walker may be in with respect to step size. It is worth noting that L_{step} is the step size taken by the particle corresponding to Brow-

nian motion when the particle is close to the cluster. This value is set before the simulation starts. What regime the particle is in depends on wheter the approximated distance between the walker and the cluster, d_{wc}, is so small that there is a chance of hitting the cluster if the particle takes a step of length L_{step}. The distance d_{wc} can be found by looking in the cell labeled with the rounded coordinates of the walker in **C**. In addition, one must take into account that what is stored is the coordinates of the centre of the particles, meaning that there must be a distance of $2r_p$ between them at least. Since one looks at the rounded off coordinates in when finding d_{wc}, one adds a 1 to make sure that the region includes all possibilities of hits.

$d_{wc} \leq 2r_p + L_{step} + 1$: The particle is in the short range regime, within hitting range of the cluster. That may be a single particle or lots of particles, and exact calculations are required for determining if the particle will hit any of the particles within the range of d_{wc}. This is done by looking at all particles stored in **B** within a square of size $2(2r_p + L_{step} + 1) + 1$ centered around the cell with indices given by the rounded coordinates of the walking particle. All non-zero elements within this square gives the location of the coordinates in **A**, so they can be extracted. Then the hit estimation calculations are performed for all particles within hitting range, and the one yielding the lowest positive step required will be the one the particle latches on to.

$2r_p + L_{step} + 1 \leq d_{wc} \leq D_{max}$: The particle is in an intermediate regime, meaning it can only move a distance of $d_{wc} - (2r_p + L_{step} + 1)$ without having to check for a possibility of a hit.

$d_{wc} = D_{max}$: The particle is in the long range regime, far away from the cluster, allowing it to move a distance of $\max(d_{w0} - R_{max} - (2r_p + L_{step} + 1), D_{max} - (2r_p + L_{step} + 1))$.

The particle will attach to the cluster iff the step performed is shorter than L_{step}. Then **A**, **B** and **C** are updated, and a new particle starts walking. A new walker is launched from the starting distance R_{start}, and this entire process is repeated until the cluster contains the desired amount of particles. At this point, the fractal dimension d_f can be calculated using a fit of the data to equation (1). This was done to evaluate and analyze the results.

3.2 Diffusion limited cluster aggregation

Diffusion limited cluster aggregation (DLCA) is the growth process where instead of starting with a seed particle and launching random walkers, one

instead starts with a density of particles over a domain, and let them all do a random walk. They will similarly to the DLA case stick together on contact, causing clusters to be formed. These cluster will again move around, and attach to other clusters until all particles are part of a single cluster.

The algortihm for DLCA has been developed for this project. It is inspired by the concepts of Kuijpers et. al. [6]. However, the variable step size determination in the off-lattice DLA algorithm has not ??? YET been implemented, as it is not as straight forward to keep track of distances between clusters and the orientation of clusters. This is partly due to the fact that all clusters will perform a random walk throughout the simulation, and may then been thought of as walkers. As with DLA, this was implemented with particles both on and off lattice. ??? To make things physical, the step length in the off-lattice simulations was calculated for each cluster according to equation (4).

The starting point of DLCA is to have particle density of interest, and distributes this over a domain. The particles are distributed randomly ???(uniformly at the moment) within the given domain. For simplicity, one uses periodical boundary conditions. The hit checks/calculations are similar to those for DLA (both on and off lattice). Particle positions are stored in a dynamical array $\mathbf{\Phi}$, as seen in figure 3 and 4d. The labels of the clusters correspond to the column number of $\mathbf{\Phi}$, and are stored in the enumerated grid $\mathbf{\Psi}$ (see figure 4b). For the off-lattice case, an extra enumerated array, $\mathbf{\Omega}$, is necessary to be able to go back and forth between the exact coordinates and mapped arrays more easily.

This storage system allows for easy checks to see if a particle has any neighbours (at the moment only nearest neighbours), which may allow for a hit. The benefit of this is that one only have to check a fraction of the total domain, limiting the computational complexity significantly. ??? This is where the hitting is explained, with the square check and everything! More specifically, the algorithm checks a square of size $4r_p + 1$ centered in the cell the particle is supposed to move to. This limits the maximum step size allowed in the program ??? calculate the limit and write about it here!. The exact coordinates of all particles within this square is extracted, and checked for hit possibility. The one hit resulting in the shortest step performed will then be the valid one ??? rephrase. Since the moving particle might be part of a cluster, the algorithm has to find the shortest valid step for every particle it contains before an actual move may take place.

In the event that a hit happens, the two colliding clusters will join together and form a new cluster. As a result, all arrays have to be updated, making

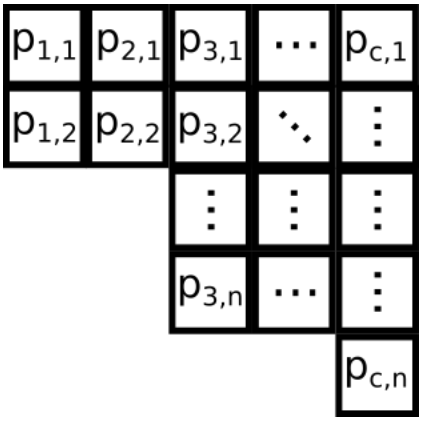


Figure 3: Schematic of the structure of the $\mathbf{\Phi}$ array. Note that the dimension is not $N \times N$, but is dynamic iand will change throughout the simulation. The subscript c indicates the label for the cluster, while n denotes the particle number within a cluster c.

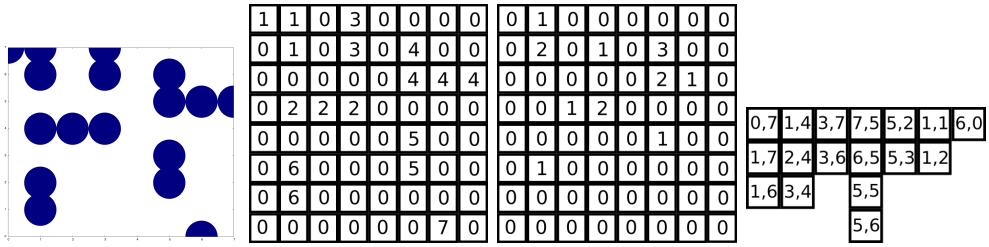
sure the labelling of the new cluster is correct. For conventional ???consistency? purposes, the new cluster will always have the label of the lowest numbered cluster comming in. This means if cluster 2 collides with cluster 6, the new cluster will keep label 2 while 6 is erased from $\mathbf{\Phi}$. All clusters with label higher than 6 will then have their label reduced by 1. This also means that all cells with label 6 in $\mathbf{\Psi}$ will be relabeled to 2, and all clusters with higher labels than 6 are reduced by 1. In $\mathbf{\Omega}$, there has to be a relabelling of the particles from the old cluster 6, so that they continue the labelling cluster 2 had before the collision.

??? Explain how when a cluster crashes into another, this will cause the cluster "crashed into" to not take a step during that time iteration.

After some while, one might have a situation similar to that shown in figure 4a. ??? elaborate on the configuration.

When a cluster moves, all particles within the cluster moves in the same distance with the same step length. An important part of the algorithm is to keep the Ψ and Ω arrays correct, even after a particle moves, as this is necessary to be able to check the correct particles for hit. ??? Write about the update process for general movement of the clusters, not related to if they hit or not. (note that this is the bug that has taken me 2 weeks to fix!)

The following is what I plan to do before Dec. 18.



(a) Example of how a configuration would look.

(b) Example of how the Ψ array would would look.

(c) Example of how the $\mathbf{\Omega}$ would look. Note that this example is for on-lattice DLCA, so it is not stricly necessary, but is is uncluded for illustrational purposes.

(d) Example of how the clusters array would look.

Figure 4: Example of how the data structures would look.

3.3 To do

Introduce physical diffusion constant, and compare cluster size distributions to what was found in [4]. Then introduce a velocity field (pretty basic to begin with), and analyze the resulting distribution in the same framework. If there is time, allow for mineral dust-sized particles in the system, and allow them to "fall out". In other words, think of the figures presented here to be in the x, y-plane, and z to be height over ground. Removing the particles will correspond to them falling down, but I will have to think about the best way to do this, while still maintaining mass conservations (introduce new small or big particles).

4 Results

The results for the on-lattice simulations can be found in figure 5. ??? The fractal dimension was calculated straight forward in this case, only looking at the relation between R and N, not using equation (1). Should probalby also update the figure with a figure of more particles, so just get it done some weekend. Should probably also fix the calculation of d_f. It is worth noting that the fractal dimension oobtained from 5 is in accordance with what was found in [1], although the method of calculation was slightly different so an exact accordance is not to be expected.

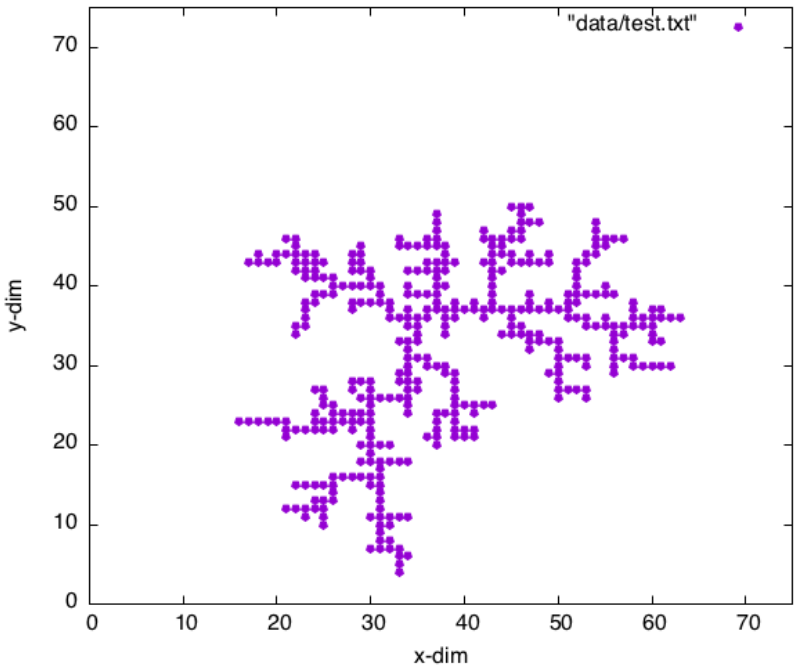


Figure 5: ??? I'm sure there could be some improvements to this. On-lattice DLA simulation of 400 particles on a 75 by 75 grid. This particular cluster has a fractal dimension of $d_f = 1.70999$.

The results for the off-lattice simulations can be found in figure 6, with corresponding regression plot in figure 10 (see equation (1)). These results fits with the results obtained by Kuijpers [6].

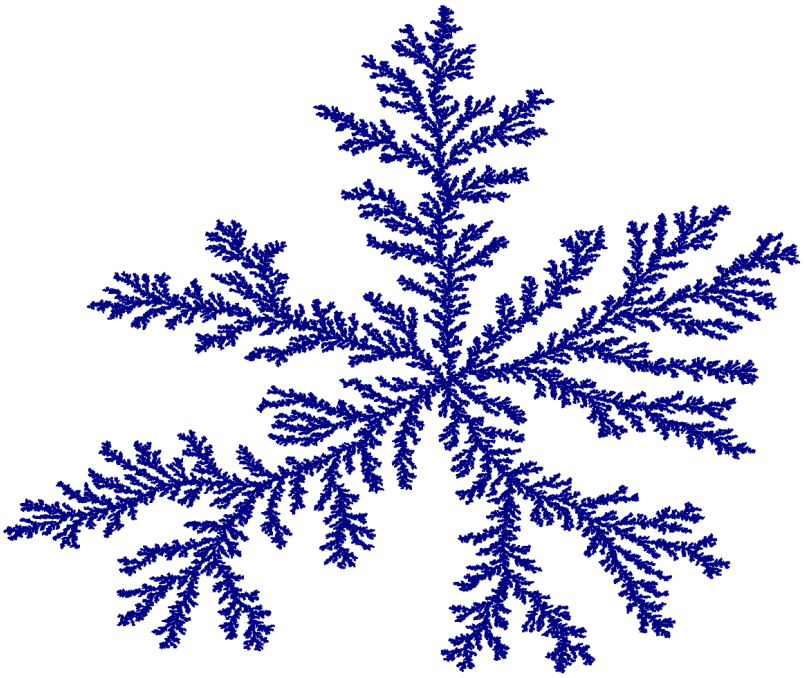


Figure 6: ??? Replace with a borderless version, so that one doesn't have to have x-ticks and x-labels etc... Off-lattice DLA simulation of 10^6 particles. This particular cluster has a fractal dimension of $d_f = 1.7248 \pm 0.006033$.

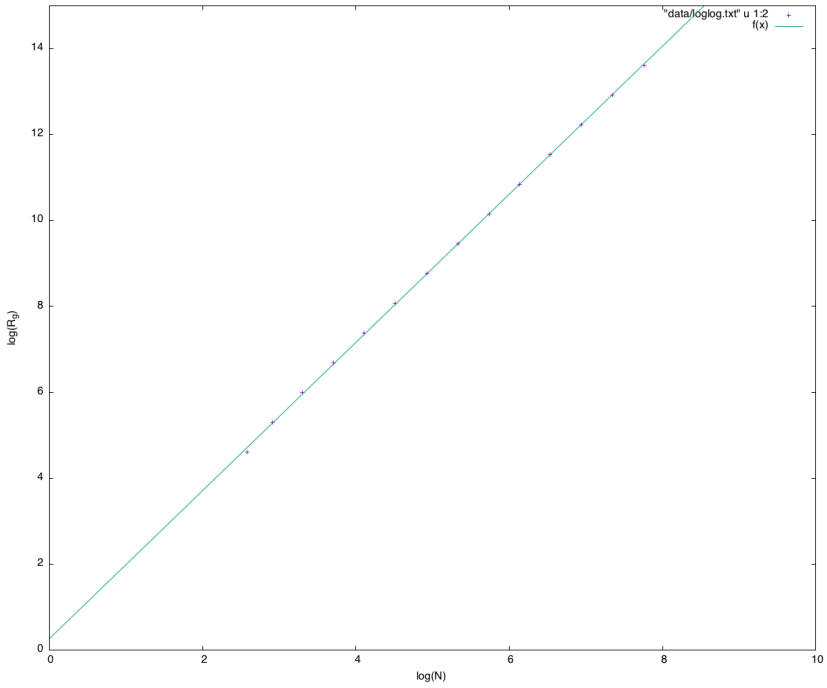
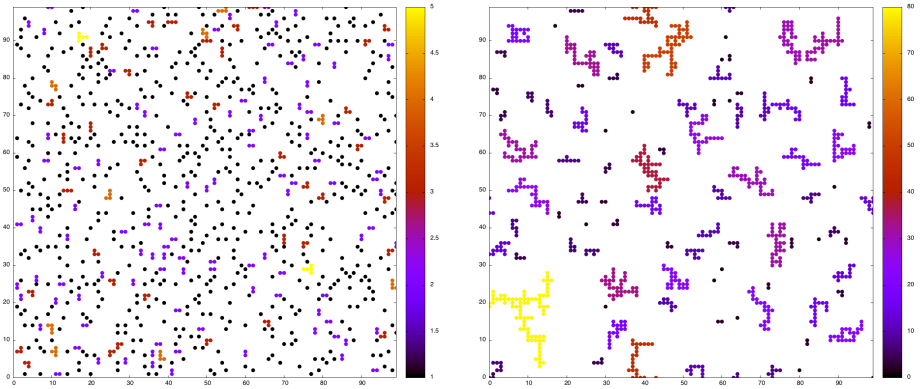
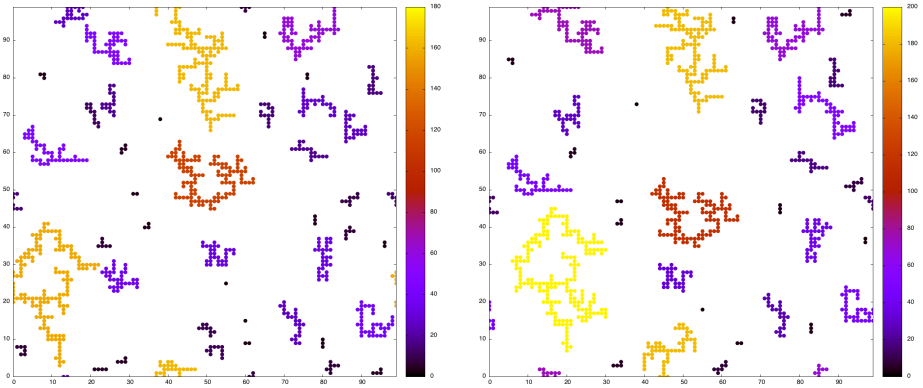


Figure 7: The log-log plot of N and R_g. From this, the fractal dimension was calculated using linear regression. ??? replace with a plot including the slope/linear approximation.



(a) How the domain looks after $t = 0$??? Change!

(b) How the domain looks after $t = 33$??? Change!



(c) How the domain looks after $t = 67$??? Change!

(d) After time $t = 100$ or whatever, must be changed at some point.

Figure 8: Sequence of how the aggregation process might look on grid. ??? has to be changed!

A References

[1] T. A. Witten and L. M. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett.*, 47:1400–1403, Nov 1981.

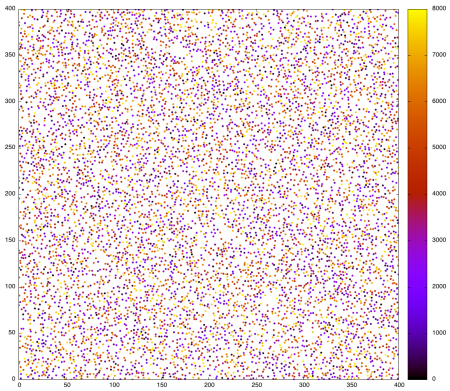
[2] Murray Eden. A two-dimensional growth process. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 4: Contributions to Biology and Problems of Medicine*, pages 223–239, Berkeley, Calif., 1961. University of California Press.

[3] Paul Meakin. Formation of fractal clusters and networks by irreversible diffusion-limited aggregation. *Phys. Rev. Lett.*, 51:1119–1122, Sep 1983.

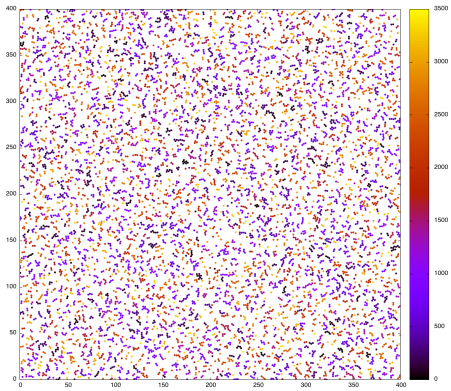
[4] Tamás Vicsek and Fereydoon Family. Dynamic scaling for aggregation of clusters. *Phys. Rev. Lett.*, 52:1669–1672, May 1984.

[5] B. V. Scarnato, S. China, K. Nielsen, and C. Mazzoleni. Perturbations of the optical properties of mineral dust particles by mixing with black carbon: a numerical simulation study. *Atmospheric Chemistry and Physics Discussions*, 15(2):2487–2533, 2015.

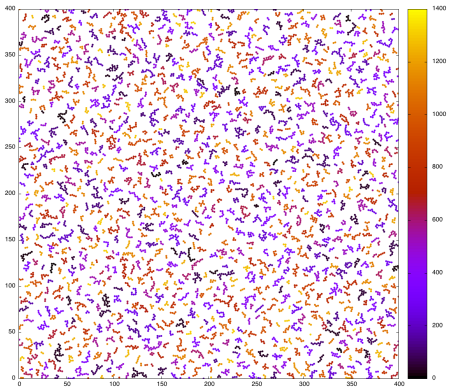
[6] Kasper R. Kuijpers, Lilian de Martín, and J. Ruud van Ommen. Optimizing off-lattice diffusion-limited aggregation. *Computer Physics Communications*, 185(3):841 – 846, 2014.



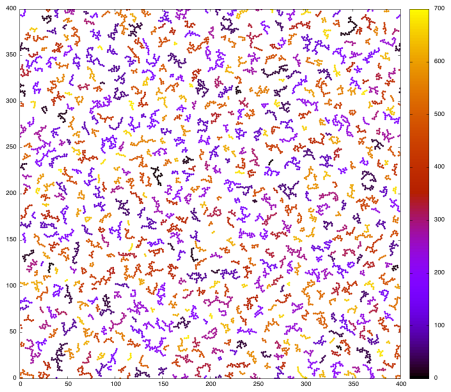
(a) How the domain looks after $t = 0.00s$??? Change!



(b) How the domain looks after $t = 0.10s$??? Change!



(c) How the domain looks after $t = 1.00s$??? Change!



(d) After time $t = 5.00s$ or whatever, must be changed at some point.

Figure 9: Sequence of how the aggregation process might look on grid. This configuration was 8000 particles on a 400×400 grid, meaning $\rho = 0.05$. This was done using steps of $dt = 0.01$, and as one would expect, the agglomeration of larger clusters is a very slow process. ??? has to be changed!

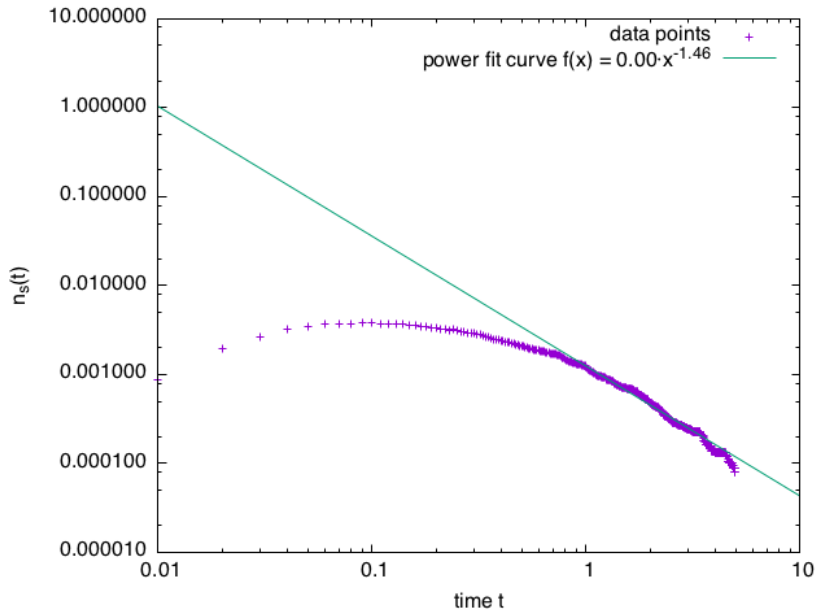


Figure 10: ??? temporary: The log-log plot of the time evolution of $n_s(t)$ for a cluster of size $s = 10$. The slope approximation to the data yields $w \approx 1.46$.