NTNU – Trondheim
Norwegian University of
Science and Technology

Modeling agglomeration of aerosols

Simen Berntsen
December 18, 2015

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

Supervisor: Alex Hansen.

1 Abstract

In the project presented here the aggregation of black carbon clusters are investigated, using an algorithmic approach. An algorithm for diffusion limited cluster aggregation is developed, and is verified by comparing it to previous methods, in particular the time evolution of the cluster-size distribution. This is analyzed with respect to the relation between the mass of the clusters and their diffusivity. Finally, a crude model for deposition is suggested where a thin layer of the atmosphere is modeled as a two dimensional plane, and the cluster flux out of this plane is estimated.

2 Preface

This thesis is the result of a one semester project in applied physics, and is a part of a Master's degree in Applied Physics at the Norwegian University of Science and Technology. The purpose of the project is to gain insight into a field, and to prepare the student for the Master thesis. First I would like to thank my supervisor Prof. Alex Hansen for all guidance throughout this project. I would also like to thank Dr. Barbara Scarnato at the University of Oslo for helping me put my research into context. Finally I would like to thank my parents and my family.

Simen Berntsen
Trondheim, Norway
December 2015

Contents

3 Introduction

Black carbon (BC) aerosols are formed from incomplete combustion of fossil fuels. Surplus carbon clump together to form clusters, which are then released into the atmosphere. The work presented here takes an algorithmic approach to calculate the agglomeration rate of BC aerosols. The motivation behind this is that emission of BC into the atmosphere can contribute significantly to both climate change and local weather effects. BC may affect the climate simply by being dark aerosols (Haywood and Shine [1]), as the aerosols absorbs radiation and heats the atmosphere (direct effect). They may also have an influence on the cloud cover, resulting in a "semi-direct" effect on heating (Koch and Del Genio [2]). Lastly, in the atmosphere, BC may act as cloud condensation nuclei in warm clouds, which increase brightness and reduce precipitation (Bond et al. [3]) A similar effect has been found in cold or mixed-phase clouds, although this is not understood as well for warm clouds [3]. The aerosols can be deposited on high reflectivity areas, for instance covered by snow/ice, to darken the surface and decrease the albedo (Hansen and Nazarenko [4]). In these areas, the BC deposition will increase melting rate, giving a positive feedback as darker ground is uncovered (Flanner et al. [5]). Recent studies (Sand. et al. [6]) suggests that emission of BC from oil fields in Northwest Russia has the greatest climatological effect on the Arctic region.

In analogy to this, one would like to investigate the effect of oil fields in subtropical desert areas with high albedo. These areas are of particular interest, due to their supplies of oil, high annual insolation, low precipitation, no known chemical or biological process removing the BC, and a high density of mineral dust in the lower atmosphere (which increases deposition rate). It is assumed that due to the previously mentioned factors, the lifetime for the BC particles on the ground may be very long. In a shorter time frame, the reduced albedo increases the absorption on the earths surface, and subsequently the lower layer of air in the atmosphere. This hot, dry air is not transported from the surface efficiently enough, which may result in heat waves during summertime. It is suspected that the reduction of the albedo in these areas will further enhance this.

In order to estimate the deposition rate in the previously described areas, one is interested in both transport and agglomeration rate of the BC particles. Agglomeration rate because pure BC clusters will have a negligible fall velocity (Flanagan and Seinfeld [7]), and transport rate because it will determine how far from the source these clusters can reach before they are deposited. In this project the growth of these pure BC clusters are investigated, using a diffusion-limited aggregation (DLA) approach. The aims of the work has been to:

- Understand the process of growth by diffusion limited aggregation.
- Develop a model for diffusion limited aggregation of clusters (DLCA).
- Investigate a crude deposition model for pure BC clusters.

To accomplish this, the DLA algorithm has been investigated, and a DLCA al-

gorithm has been developed. DLA is an algorithmic approach to an irreversible growth process where the limiting factor is the amount of mass available in the system. A number of different growth models have been proposed, the original by Witten and Sander [8]. This model builds on the Eden model, first proposed by Eden [9], but with a random walker determining the aggregation. A more dynamic model for agglomeration of moving clusters was described by Meakin [10]. In the work presented here, a similar model was developed and analysed using the scaling relation suggested by Family and Vicsek [11]. The diffusivity of the clusters is investigated and compared to the findings of Meakin et al. [12]. Some of the physical properties of the system have been investigated, in particular the mass dependence of the diffusivity of the clusters, and qualitatively how diffusivity affects the deposition rate. The results are then evaluated and compared with previous works. Finally a crude model for estimating the deposition rate is presented, and the results from these simulations are discussed.

All the code developed and used in this project can be found in [13].

4 Theory

Since this project takes an algorithmic approach, the theory explaining the validity of this approach is presented. The theoretical framework for analyzing the results is also discussed briefly.

4.1 Diffusion limited aggregation and fractal dimension

As described in section 3, the DLA algorithm describes the irreversible growth process determined by random walkers attaching to a cluster. Einstein showed that if the steps Δx taken are sufficiently small, then random walks and Brownian motion are related [14] through

$$D = \frac{(\Delta x)^2}{2\Delta t}, \tag{1}$$

where D is the diffusion constant (also called diffusivity), Δx is the step taken by the walker, and Δt is the time it takes the walker to perform one step. Equation 1 explains how the random walkers proposed by Witten and Sander [8] can be part of a diffusion process. The cluster formed by this process displays the branching, self-similar structure found in fractals (see Barabási and Stanley [15]). Without giving a full introduction to fractals (see for instance chapter 10 in Strogatz [16]), the Hausdorff dimension (also called fractal dimension) can be calculated for aggregates with these properties. There are several ways to do this, either by calculating the density-density correlation function for the aggregate, or by the so-called radius of gyration method. The latter is the most popular, and is the method used in this project. It is defined from the radius of gyration

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

where R_g is the radius of gyration, N is the number of particles in the cluster, and r_i is the distance between the center of mass of the cluster to particle number i. In other words, the radius of gyration is the average distance between particles in the cluster and the center of mass. Now this may be used to estimate the fractal dimension d_f of the cluster, via (Meakin [17])

$$N \sim R_g^{\beta}. \tag{3}$$

N is the number of particles in the cluster, R_g is the radius of gyration, and $\beta = \frac{1}{d_f}$ gives the fractal dimension of the cluster. The conventional way of determining the fractal dimension is by calculating the slope of $\ln(R_g)$ as a function of $\ln(N)$.

4.2 Aggregation of clusters and Family-Vicsek scaling

The possibility for collision between moving clusters means that the self-similar structures found in aggregates formed by DLA are not found in aggregates formed by DLCA. The fractal dimensions are no longer the most interesting feature, but instead one analyzes the cluster-size distribution $n_s(t) = N_s(t)/L^2$ describing the amount of clusters $N_s(t)$ of size s at time t in a domain of size L. This is done to understand the temporal evolution of the system. Family and Vicsek [11] found that $n_s(t)$ decays as a power law of the form

$$n_s(t) \sim t^{-w} s^{-\tau} f\left(\frac{s}{t^z}\right), \tag{4}$$

where w, τ and z are exponents to be determined from the simulations. The function $f(x)$ is a cut-off function expected to behave as $f(x) \approx 1$ for $x << 1$ and $f(x) << 1$ for $x >> 1$.

With moving clusters, one should treat equation (1) with caution. One suspects that the diffusivity of a cluster depends on the cluster mass M_s. For identical particles, the mass of a single particle is taken to be unity, allowing for the substitution $M_s \to s$, where s is the number of particles in the cluster. An ansatz is made for this relation, connecting D to the mass M_s of the cluster (Meakin et al. [12])

$$D = D_0 s^{-\gamma}. \tag{5}$$

In equation (5), D_0 is a constant, s is the number of particles in the cluster, and γ is an exponent determining the influence the mass of the cluster has on the diffusivity.

5 Diffusion limited aggregation

In order to be able to design an algorithm for diffusion limited cluster aggregation (DLCA), some familiarity to the field is required. As an introduction and preparation, the case of regular DLA was investigated, both on and off lattice. The main interest of section 5 is to prepare for the work done in section 6, where a more general DLCA algorithm has been developed. Although it is previously stated that one is interested in the agglomeration rate in this project, this is not the case for the DLA studies performed. However, a lot of interesting aspects that are useful to the DLCA algorithm are uncovered.

The fundamental algorithm of DLA, as described by Witten and Sander [8] is as follows; a seed particle (the particle we follow) is placed at the center of a domain of interest. 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. This will form a rigid cluster, which grows as more walkers hit it. The diffusion takes place in a two dimensional domain for all simulations.

5.1 On-lattice DLA

5.1.1 Method

For the on-lattice case, the algorithm is pretty straight forward, as described in the beginning of section 5. Thus it will only briefly be discussed here, with the main focus on the off-lattice version. Special for the on-lattice case is that the particle can only move in discrete steps usually set to $L_{\text{step}} = a_l$ (where a_l is the lattice constant), and in discrete 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 particle will always start at a random (uniformly distributed) point on the perimeter of the grid. If it takes a step causing it to move outside of the grid, it will restart somewhere else on the perimeter. Although this cannot be done without some loss of generality ("killing" a particle that walks out of the grid will change the probability of ending up in some locations by a very small fraction), it is still done to increase the simulation speed. Since the on-lattice simulation is of less interest compared to the off-lattice case, only a crude on-lattice simulation was performed.

5.1.2 Results

The results for an on-lattice simulations can be found in Figure 1. In this particular simulation, $N = 400$ particles were placed on a square lattice of size $L = 75a_l$. For simplicity a_l is set to 1 (unity), and $L_{\text{step}} = a_l$. While more particles would have been beneficial to ensure the validity of equation (3), Figure 1 illustrates the branching structure of the DLA cluster, which can be found in clusters at multiple scales due to the self-similarity. The fractal dimension obtained from the simulation shown in Figure 1 is $d_f \simeq 1.71$ and is in accordance with what was found in [8]. The method for calculating the fractal dimension was slightly different, so an exact match is generally not expected.

Figure 1: On-lattice DLA simulation of 400 particles on a 75 by 75 grid. The Figure illustrates the characteristic branches found in the fractal structures created by DLA processes. This particular cluster has a fractal dimension of $d_f = 1.71$.

5.2 Off-lattice DLA

5.2.1 Method

In off-lattice DLA simulations, the particle is no longer restricted to move along any axis, or to take discrete steps. This means it can move freely in an approximately continuous space, and in any direction. As a consequence, the algorithm is required to be more advanced 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. [18]. By defining a default step length L_{min} which corresponds to diffusion (equation (1)), one may take longer steps (L_{step}) if the walking particle is far from the cluster to speed up the simulation. Additionally, this algorithm uses an on-lattice approximation for storing the positions of the particles in the cluster. Therefore the algorithm is more RAM dependent than other algorithms, but with the benefit of reducing the calculation required to identify sites viable for collision. Like in section 5.1.2, the distance between two lattice points a_l is set to unity. In analogy to what was the case in section 5.1, the particle starts along a circle of radius R_{start} from the cluster, and will be "killed" if it wanders too far from the cluster. This limit is set

to $R_{\text{kill}} = 5R_{\text{cluster}}$, where R_{cluster} is the radius of the smallest circle spanning the cluster. This is once again done to speed up the simulations, at the cost of changing the probability to end up in some locations ever so slightly.

As one goes from discrete to continuous space, the algorithm must calculate if a collision is taking place, as opposed to checking if neighboring lattice points are occupied. With the particles no longer restricted to be on lattice points, floating point coordinates define their position. All particles are defined from their center, meaning one has to take into account their extent. The method devised for calculating if a collision occurs is illustrated in Figure 2. Cartesian coordinates were used, and note that the exact coordinates of particles eligible for hits (x_p ,y_p) and the walking particle (x_0, y_0) are known at the point in the simulation when this calculation is performed. Why these are known is explained in the section describing the ($d_{wc} \leq 2r_p + L_{\text{step}} + 1$)-regime on page 9. A step direction $\alpha \in [0, 2\pi]$ (defined as seen Figure 2) is drawn from a uniform random distribution for each particle at the start of every iteration. The distances a and b are found from geometric inspection of Figure 2, giving

$$a = x_p - (x_0 + L_{\text{hit}} \cos(\alpha)) \tag{6}$$
$$b = y_p - (y_0 + L_{\text{hit}} \sin(\alpha)). \tag{7}$$

Inserting (6) and (7) into $d_p^2 = a^2 + b^2$ and rearranging gives

$$\begin{aligned} d_p^2 =& L_{\text{hit}}^2 + 2\left[(x_0 - x_p)\cos(\alpha) + (y_0 - y_p)\sin(\alpha)\right] L_{\text{hit}} \\ &+ \left[x_0^2 + x_p^2 + y_0^2 + y_p^2 - 2(x_0 x_p + y_0 y_p)\right]. \end{aligned} \tag{8}$$

This can be written on the form

$$AL_{\text{hit}}^2 + BL_{\text{hit}} + C = 0, \tag{9}$$

with coefficients defined by

$$A = 1 \tag{10}$$
$$B = 2\left[(x_0 - x_p)\cos(\alpha) + (y_0 - y_p)\sin(\alpha)\right] \tag{11}$$
$$C = (x_p - x_0)^2 + (y_p - y_0)^2 - d_p^2. \tag{12}$$

Equation (9) may be solved as a usual polynomial of second order, which means it may or may not have a real solution. The algorithm is thus designed to take all possible situations into account. Only if $L_{\text{hit}} > 0$ *and* $L_{\text{hit}} < L_{\text{min}}$ will L_{step} be set equal to L_{hit} and the particle will collide. All other solutions of (9), will result in $L_{\text{step}} = L_{\text{min}}$. With α and L_{step} known, calculating the position of the particle after the step is performed is straight forward.

With the new collision calculation routine, the concept of this new algorithm is as follows: The program uses three arrays (illustrated in Figure 3), which together keeps track of all information of interest. The first one, $\mathbf{A}$, is an $N \times 2$ array (N

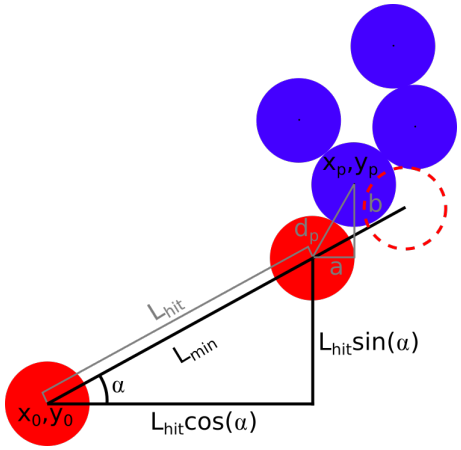
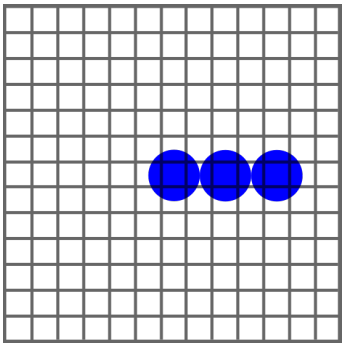


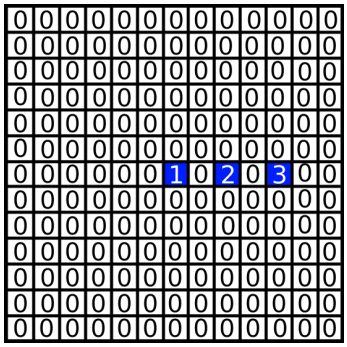
Figure 2: Schematic of the hit calculation for off-lattice simulations. Here, the red circles are the positions of the walking particle, and the blue are particles already in the cluster. The walking particle is launched from position x_0, y_0 in the α direction with the horizontal plane. The dashed red line indicates where the walking particle would end up, if it was to take a full step of length L_{min}. 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 $d_p = 2r_p$ is the diameter of the particles, while a and b are geometrical distances.

is number of particles), in which the exact coordinates of the particles are stored. These coordinates can then be rounded off to integer values, which will correspond to a cell number in $\mathbf{B}$, an $L \times L$ (L is the size of the approximation grid) array in which all the elements are either 0 or the label of the particle. See Figure 3a and 3b for illustrations 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 $L \times L$), stores the rounded off distance from each cell to the closest particle, as shown in Figure 3c. 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. 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).

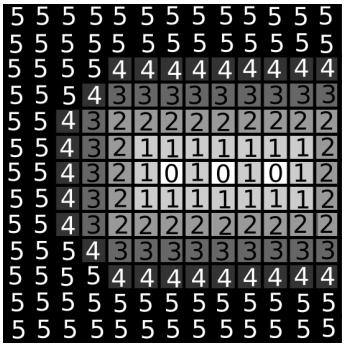
Throughout the process, the walking particle will take steps depending on its distance from the cluster. There are two criteria that determine the different regimes the walker may be in with respect to step size. If the distance between between the walker and the cluster $d_{wc} \leq 2r_p + L_{\text{step}} + 1$, with r_p being the radius of the particle, then the particle is considered to be close to the cluster. Alternatively, if $d_{wc} = D_{\text{max}}$ it is considered to be far away. What regime the particle is in depends on whether the approximated distance between the walker and the cluster 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



(a) Figure of the example configuration with the grid of size L displayed.



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



(c) Illustration of the array with the rounded off distances, here $D_{\max} = 5$. Knowing what cell the walker is in, it is easy to determine its distance to the cluster.

Figure 3: Illustration of a example configuration of particles, and how each array work. The seed particle is in all cases placed in a 13×13 array, at position (7.5, 7.5). For simplicity, the other particles are not random walkers in this illustrational case, but move in from right to left, along the horizontal axis.

the rounded coordinates of the walker in **C**. In addition, one must take into account that what is stored are the coordinates of the center 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 when finding d_{wc}, one adds a 1 to make sure that the region includes all possibilities of hits.

$d_{wc} \leq 2r_p + L_{\mathbf{step}} + 1$
: 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 with sides of length $2(2r_p + L_{\text{step}} + 1) + 1$ centered around the cell with indices given by the rounded off coordinates of the walking particle. All non-zero elements within this square gives the label 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.

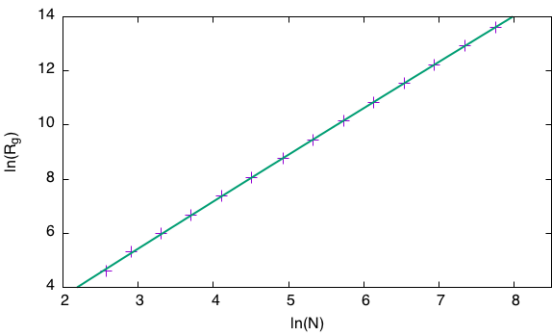
$d_{wc} = D_{\mathbf{max}}$
: The particle is in the long range regime, far away from the cluster, allowing it to move a longer distance to speed up the simulation. The resulting step length is determined by $L_{\text{step}} = \max(d_{w0} - R_{\max} - (2r_p + L_{\text{step}} + 1), D_{\max} - (2r_p + L_{\text{step}} + 1))$. The distance d_{w0} is the distance from the walking particle to the seeding particle, and $R_{\max}$ is the largest distance between any particle in the cluster and the center of mass.

The particle will only attach to the cluster if the step performed is shorter than L_{step}. Then **A**, **B** and **C** are updated, and a new particle starts walking. A new walker

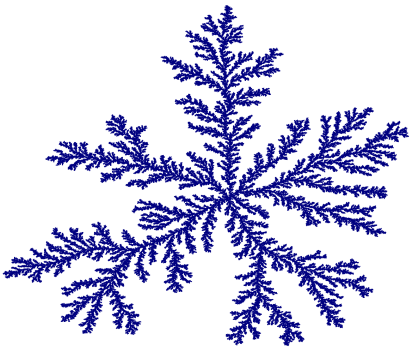
is launched from the starting distance R_{start}, and this entire process is repeated until the cluster contains the desired amount of particles. At this point, the fractal dimension d_f can be calculated using a fit of the data to equation (3). This was done to evaluate and analyze the results.

5.2.2 Results

The results for an off-lattice simulation using $N = 10^6$ particles on a $L = 1.3 \times 10^4$ domain can be found in Figure 4b, with corresponding regression plot in Figure 4a (see equation (3)). These results fits the results obtained by Kuijpers et al.[18], but are more importantly in accordance with the results of Witten and Sander [8]. One can recognize the branching structure of Figure 1, although in Figure 4b the particles are not confined to a grid. As demonstrated by Kuijpers et al., this is a very fast version of the DLA algorithm, which suggests it is a good starting point for developing a DLCA algorithm.



(a) The log-log plot of N and R_g. From this, the fractal dimension was calculated using linear regression. The purple points are the simulated data, while the green line is the fitted curve with a slope $\simeq 1.72$.



(b) The resulting structure after letting 10^6 particles grow a cluster in an off-lattice DLA simulation.

Figure 4: Off-lattice DLA simulation of 10^6 particles. This particular cluster has a fractal dimension of $d_f = 1.725 \pm 0.006$.

6 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 starts with a density of particles over a domain, and let them all walk randomly (see Meakin [10]). They will similarly to the DLA case stick together on contact, causing rigid clusters to be formed. These cluster will again move around, and attach to other clusters until eventually all particles are part of a single cluster. This may be done by Monte Carlo simulation, such as in [10], where one cluster is selected at random, and that cluster does one step of a random walk. This is repeated for every iteration in time. An advantage of the Monte Carlo method is that one can scale the time step Δt according to the number of particles s in the cluster selected to be moved, and the total number of particles N_0. This is done by incrementing the time $\Delta t = \frac{s}{N_0}$ for every time a cluster is moved.

The method presented in the following section differs from the Monte Carlo simulations by moving every cluster for each time step (they are however moved sequentially). This is done because it is expected that the concentration of BC particles will decay rapidly with time. One can only expect conditions to be relatively stable in the short term regime, where the Monte Carlo method is not valid. This method ensures that all particles will move for each time step. In addition, the program devised allows for primary particles of different sizes, however all simulations are run with identical particles in this project. Once the particles collide, they will form clusters of varying size.

6.1 Method

The DLCA algorithm presented in the following section has been developed during this project. It is based on the work of Meakin [10] and inspired by some of the concepts of Kuijpers et. al. [18], by using an on-lattice approximation (referred to as supporting arrays in the following) to limit the domain of interest to a subdomain around the particles of interest. As with DLA, this was implemented with particles both on and off lattice, and were simulated for two dimensions.

The starting point of DLCA is to have a particle density of interest, and distribute this over a domain. The particles are distributed randomly from a uniform distribution within the given domain. For simplicity, one uses periodic boundary conditions. The collision checks and calculations are similar to those for DLA (both on and off lattice). To account for the relation between mass and diffusion length, the step length L_{step} is calculated for every cluster at every iteration, according to equations (1) and (5). Exact particle positions $p_{n,c}$ are stored in a dynamical array $\mathbf{\Phi}$, as seen in Figure 5 and 6b. The clusters are labeled, corresponding to the column number of $\mathbf{\Phi}$, and are stored in the enumerated support array $\mathbf{\Psi}$ (see Figure 6d). This label corresponds to the column and cluster number (c in $p_{c,n}$) of the $\mathbf{\Phi}$ array. This is all that is needed for on-lattice simulations of DLCA. For the off-lattice case, one

must know the internal labeling in each cluster. This is to be able to extract the exact coordinates of the particles for doing the calculation to check for collisions, and is completely analogous to what was done in section 5.2.1. An extra support array $\mathbf{\Omega}$ is introduced, which keeps track of the labeling within each cluster, starting from the original particle. This enumeration of the particles corresponds to the row (n in $p_{c,n}$) in $\mathbf{\Phi}$ where their coordinates are stored. In other words, c (cluster label) and n (internal label of particle within that cluster) are found in $\mathbf{\Psi}$ and $\mathbf{\Omega}$ respectively, in the cells pointed to by the rounded down coordinates. These arrays are introduced to be able to go back and forth between the exact coordinates and supporting arrays easily.

The idea behind going back and forth between the exact coordinates and the supporting arrays is that this can help the program to know the orientation of the clusters. The rounded down coordinates from $\mathbf{\Phi}$ point towards a cell in $\mathbf{\Psi}$ and $\mathbf{\Omega}$. The numbers stored in those cells will be the c and n respectively, pointing towards $p_{c,n}$, as seen in Figure 5. Due to 0 being chosen as the indicator for empty cells in $\mathbf{\Psi}$, the c index of $p_{c,n}$ starts at 1 (see Figure 6d). Since $\mathbf{\Psi}$ keeps track of what cells are empty and not, the same is not necessary for $\mathbf{\Omega}$, meaning the internal labeling of the particles may start at 0 (see Figure 6c).

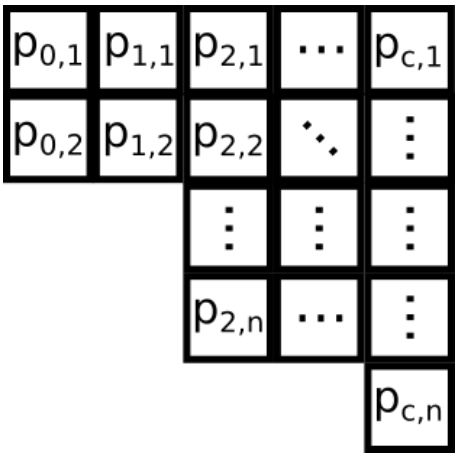
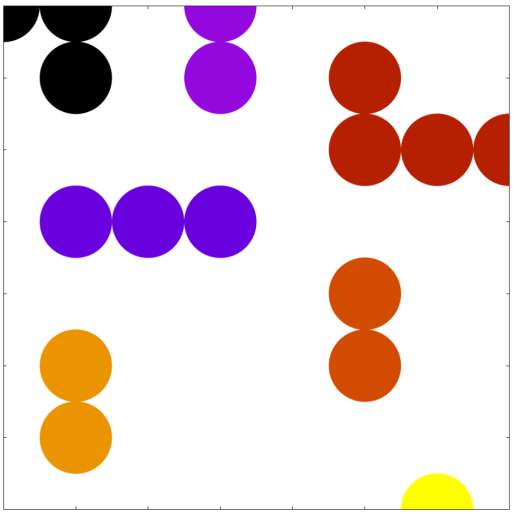


Figure 5: Schematic of the structure of the $\mathbf{\Phi}$ array. Note that the dimension is not $N \times N$, but is dynamic and will change throughout the simulation. Initially, all the particles are single-particle clusters. The subscript c indicates the label for the cluster, while n denotes the particle label within a cluster c.

A storage system like this allows for easy checks to see if a particle has any neighbors, which may allow for a collision. The benefit of this is that one only has to check a fraction of the total domain, limiting the computational complexity significantly. More specifically, given the location of a particle, a step length L_{step} and direction α, the algorithm calculates where the particle would move had there been no other clusters. It will then check a square with sides of length $4r_p + 1$ centered in the cell the particle is attempting to move to. How this is done is illustrated in Figure 7. As a consequence, the maximum step size allowed in the program to avoid overlooking potential collisions is limited. A step is only valid if $L_{\text{step}} < 2r_p + 2r_{\min}$, where r_p is the radius of the particle attempting the move, and $r_{\min}$ is the radius of the smallest



(a) Example of how a configuration would look.

0,7	1,4	3,7	7,6	5,2	1,1	6,0	
1,7	2,4	3,6	6,6	5,3	1,2		
1,6	3,4		5,6				
			5,7				

(b) Example of how the $\mathbf{\Phi}$ array would look. Components are the coordinates of the particles shown in 6a. These coordinates are stored in label (c, n).

0	1	0	0	0	0	0	0
0	2	0	1	0	3	0	0
0	0	0	0	0	2	1	0
0	0	1	2	0	0	0	0
0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c) Example of how the $\mathbf{\Omega}$ would look. The components are the internal labeling of particles n within a given cluster c.

1	1	0	3	0	0	0	0
0	1	0	3	0	4	0	0
0	0	0	0	0	4	4	4
0	2	2	2	0	0	0	0
0	0	0	0	0	5	0	0
0	6	0	0	0	5	0	0
0	6	0	0	0	0	0	0
0	0	0	0	0	0	7	0

(d) Example of how the $\mathbf{\Psi}$ array would look. The components are the respective cluster labels c for the particles in the system.

Figure 6: Example of how the data structures could look in an on-lattice simulation of DLCA. The colors of the particles are determined by what cluster they are a part of. Note especially how since the enumeration of the clusters in $\mathbf{\Psi}$ starts at 1 (0 representing an empty cell), the $\mathbf{\Phi}$ array is 1-indexed. By knowing both c and n, one can go in back and forth between the storage locations in all of the arrays presented here.

particle in the system. This was done to speed up the simulation for medium sized clusters, and has little effect on simulations where the particles take small steps. Had one been interested in allowing longer steps, one could for instance check a larger square centered in the starting location of the particle.

The exact coordinates of all particles within this $4r_p + 1$ sized square are extracted, and checked for collision possibility, as shown in Figure 7. The shortest step that

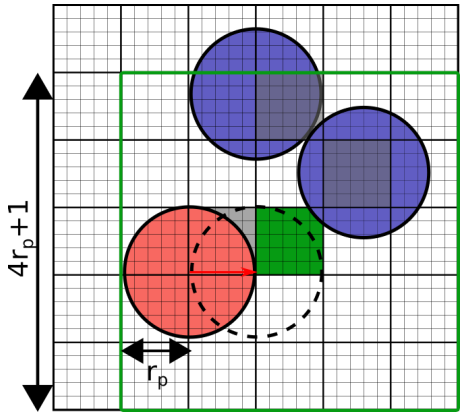


Figure 7: Illustration of how the algorithm extracts the labels to the exact coordinates. The particle about to be moved is marked with red (dashed black line indicates destination), while particles eligible for collisions are marked blue. Shaded cells represent the approximation to the support arrays (and via these the location in $\mathbf{\Phi}$), while the green cell is the one the particle attempts to move to. Note that the coordinates are rounded down, but this is just a choice made for consistency. Step direction α and step length L_{step} (see red arrow) is for simplicity set to $\alpha = 0$ and $L_{\text{step}} = r_p = a_l$, where a_l is the lattice constant. The domain in which particles has to be checked for collision eligibility is reduced to the subdomain with the green outline, as the moving particle may only collide with particles within this domain.

will result in a collision is the one taken by the particle. Since the moving particle might be part of a cluster, the algorithm has to find the shortest valid step for every particle in the cluster before an actual move can happen.

In the event that a collision happens, the two colliding clusters will join together and form a new cluster. All arrays have to be updated, making sure the labeling of the new cluster is correct. To be consistent, the new cluster will always have the label of the lowest numbered cluster coming in. This means if for example 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. Similarly, 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 relabeling of the particles from the old cluster 6, so that they continue the internal labeling cluster 2 had before the collision.

As a result of this, the cluster that has been hit will not perform a step during that time iteration. This is a clear weakness of the model, since the system loses some of its energy (net movement) as a result. The immediate alternative would be to have some particles moving twice during a time step, so this solution is taken with the argument that the loss of energy may be neglected for large clusters, since they will move so slowly that the difference is negligible. In addition, larger clusters will tend to have a lower label as a result of this label convention, meaning that the large clusters would perform the steps before smaller clusters. The magnitude of the loss will ultimately depend of number of particles within a cluster crashed into, so large clusters with lower labels will neglect some of this. As a whole, this is expected to have a minor impact on the cluster-size distribution and be affected by the mass-diffusivity relation (equation (5)).

After some time, one might have a situation similar to that shown in Figure 6a, which is included for illustrational purposes. In Figures 6b, 6d and 6c, one can see the corresponding configurations in the data structure, and how one can move back and forth between exact coordinates and rounded down cells.

When a cluster moves, all particles within the cluster moves in the same distance with the same step length. A critical part of the algorithm is to keep the Ψ and Ω arrays correct, as these are used when checking the correct particles for hit. As cells left empty after movement must be nulled out, one has to keep track of all cells moved to and from when updating. This is a somewhat computationally intensive maneuver, however it is critical to having the algorithm working as intended.

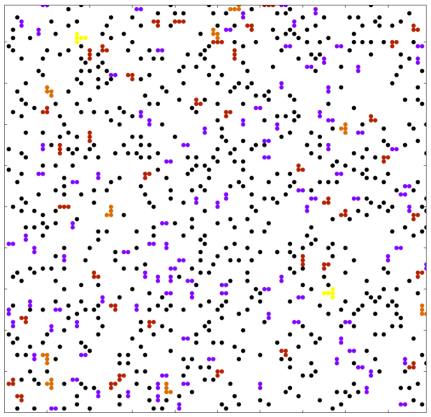
The program will end either when all particles are in a single cluster, or when a specified number of time steps has been simulated. The cluster-size distribution is calculated for every time step, for a fixed value of s (amount of particles within a cluster), and similarly for every s at a fixed time t.

6.2 Results and discussion

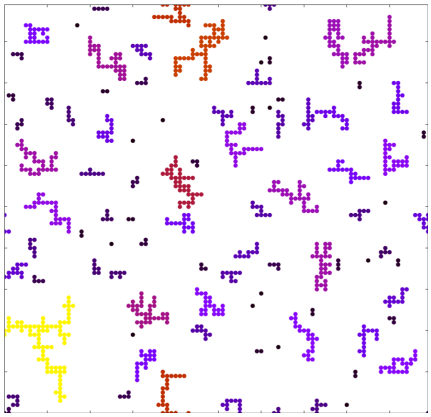
The main purpose of this section is to verify the method used, as it differs from previous works. The behavior of the cluster-size distribution function $n_s(t)$ should however be similar, independent of which method was used. Thus the main focus of this section is to evaluate the method used in this project, by comparison of the cluster-size distribution to the work done by Meakin et al. [12] and Family and Vicsek [11]. This is for the latter also done in relation to the mass-diffusion relation (5).

By performing mass-independent on-lattice DLCA simulations using the previously described algorithm, one may generate clusters as shown in Figure 8, where several configurations are plotted to display the time evolution of the system. Like for the on-lattice DLA simulations, the results obtained by the discrete version is of less importance than those obtained by the off-lattice version, and was mostly performed as an introduction to the problem. Additionally, an important point to make is that doing the discrete analysis by Monte Carlo simulations will allow for inspection of the mass-diffusivity relation, as one has the freedom of scaling the time incrementation Δt (as described in section 6, see [12] for details) by the size of the cluster moved. This is not possible when moving every cluster per time iteration in an on-lattice simulation, as both Δx and Δt is fixed. Consequently, the comparison to previous works was done for the off-lattice simulations where Δx could be scaled according to equation (5).

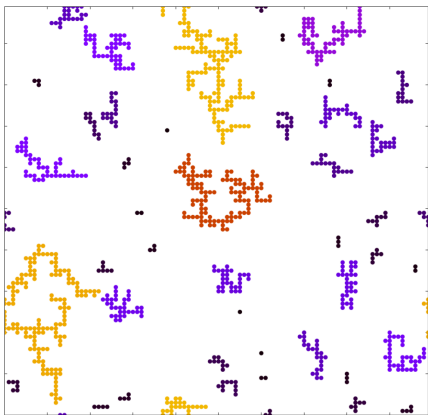
All the off-lattice simulations were done using $N = 10^4$ particles on a $L = 400$ lattice, giving a density of $\rho = 0.0625$. In Figure 10 some of the configurations of off-lattice DLCA simulations are presented, from a simulation using $\gamma = 1$. Comparison to Figure 8 reveals the differences between discrete and continuous movement, although the cluster-size distribution will in principle be independent of the method used (given the opportunity to scale either Δx or Δt). Figure 10



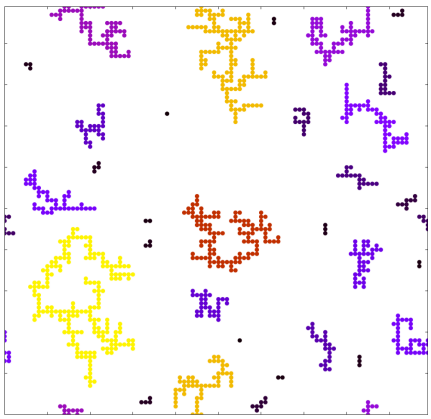
(a) Configuration of particles in the domain after $t = 0$.



(b) Configuration of clusters in the domain after $t = 33$.



(c) Configuration of clusters in the domain after $t = 67$.



(d) Configuration of clusters in the domain after $t = 100$.

Figure 8: Sequence of how the aggregation process might look in on-lattice simulation of $N = 1000$ particles on a $L = 100$ lattice. The diffusion is independent of mass ($\gamma = 0$). For the on-lattice simulations, the particles may start out as part of a cluster, but this is not the case for the off-lattice simulations.

also displays how slow the diffusion process of larger clusters will be. The mean cluster-size $S(t)$ can be found in Figure 9. As shown by Family and Vicsek [11], $S(t) \sim t^z$ is expected to diverge for large t. In [12], the slope z for $\gamma = 1$ simulations was found to be 0.42, equal to what was found in Figure 9. Comparing Figure 9 to Figure 2 in [12], one can see that the slope fits very well.

It is of great interest to study the cluster-size distribution as a function of time. For this project, the goal is ultimately to determine the agglomeration rate, which suggests the time-dependency of the cluster aggregation is important. Thus the time evolution has been investigated and compared to the results found in [11]. In the original paper, only the mass-independent diffusivity case was investigated (corresponding to $\gamma = 0$), however the dependence of γ (see equation (5)) is believed to be of importance to the agglomeration rate and has therefore been studied here. The results can be found in Figure 11, where the cluster-size distribution was investigated for a fixed cluster-size $s = 10$, and simulated up to $t = 50$.

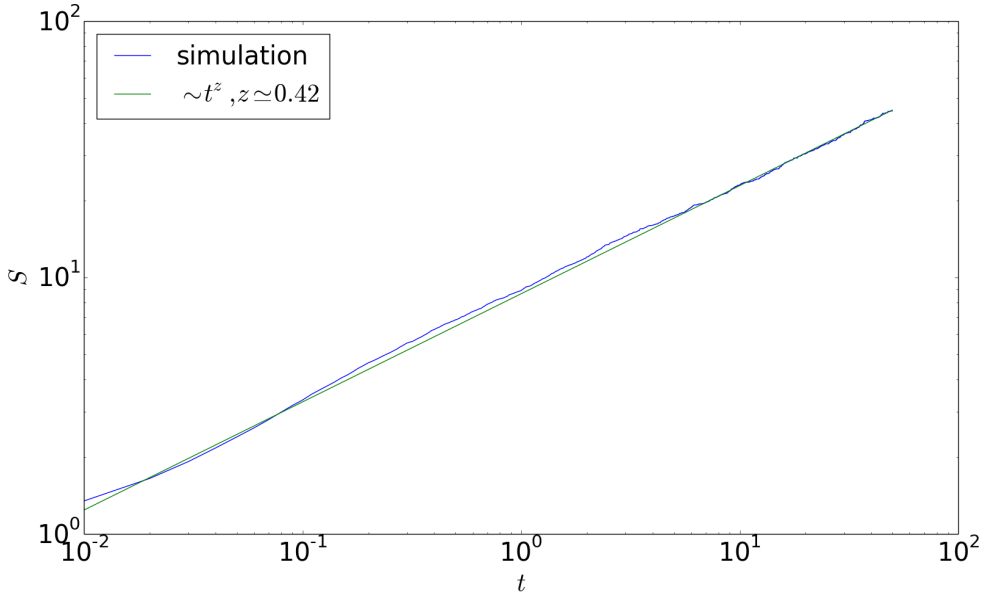
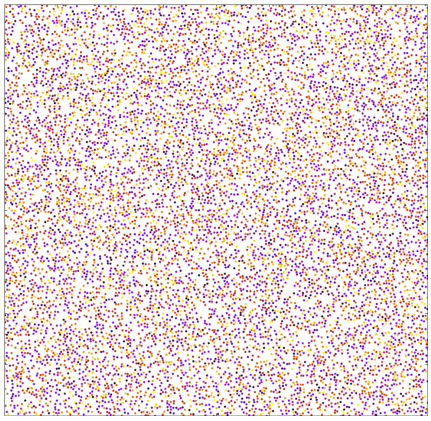


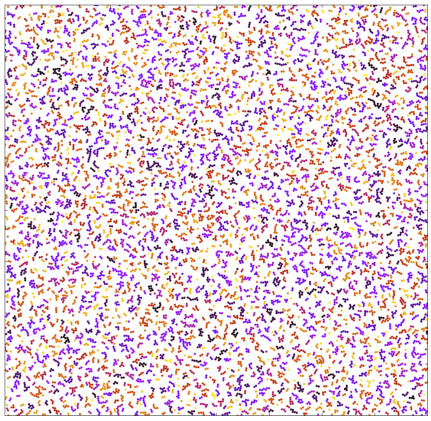
Figure 9: The mean cluster-size S as a function of time. In this off-lattice simulation, $\gamma = 1$.

In Figure 11 the time dependence of $n_s(t)$ (equation (4)) is plotted for different assumptions about how the mass of the particles affects the diffusivity (related through equation (5)). The decay of $n_s(t) \sim t^{-w}$ was determined by doing a least mean square regression on the data from the simulations. A fixed value of γ was set at the beginning of each simulation. Furthermore, it is of interest to investigate how the time decays of $n_s(t)$ is related to γ, and how this affects w. The results for the simulations investigating this are presented in Figure 11, where one clearly sees that w decreases as γ increases. This is to be expected, as heavier clusters move more slowly than lighter ones, and will tend to be more preserving of their size. This manifests itself in the broadening of the curve, an effect also seen in Figure 10d.

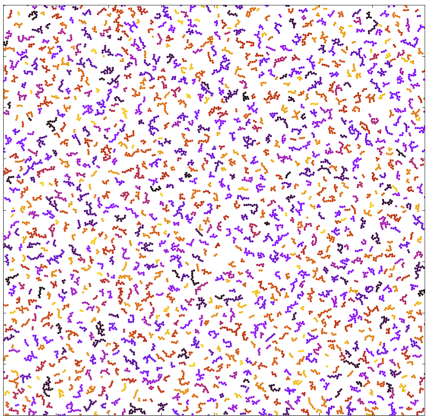
Comparing the time-behavior of the mass-diffusivity independent simulations to the results of Family and Vicsek, one sees that $w \simeq 3.90$ differs from $w \simeq 1.7$. This is suspected to be due to the difference in the starting densities, as this allows for more collisions, and thus a more rapid decay of the number of clusters of size $s = 10$. It does also appear to be a significant amount of noise in the results in Figure 11, which could be reduced by doing more simulations and averaging the results. The curve fitting is also susceptible to the range of the data it is fitted to, so an exact accordance can not be expected. The overall trends are thus either as expected, or can be explained. It would nevertheless be beneficial to run a simulation with the exact conditions explained in [11]. An in-depth analysis was not performed, but a curve of slope $w = 1.7$ was plotted along with the results from the simulations in Figure 12. The results follow the trend of the curve approximation, although with some deviations. This was included to prove that the results agrees with previous works, given the same inital conditions.



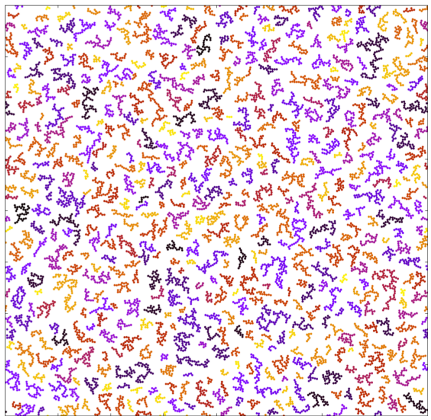
(a) Configuration of particles in the domain after $t = 0.00$.



(b) Configuration of clusters in the domain after $t = 0.10$.



(c) Configuration of clusters in the domain after $t = 1.00$.



(d) Configuration of clusters in the domain after $t = 5.00$.

Figure 10: Sequence of how the aggregation process might look for an off-grid simulation where $\gamma = 1$. This configuration was $N = 10^4$ particles on a $L = 400$ grid, meaning $\rho = 0.0625$. This was done using steps of $dt = 0.01$, and as one would expect, the agglomeration of larger clusters is a very slow process.

Similarly, the cluster-size distribution was studied for fixed times, determining the dependence on s, for multiple values of γ. The results can be found in Figure 13. In [11], only the $\gamma = 0$ case was investigated, while the γ-relation was studied in [12]. In the latter, the author does not state at what value t is fixed, making it difficult to reproduce exactly. This difficulty is increased by the suspected differences in time scaling, due to the differences between Monte Carlo simulations and the method used for this project. Some of the behavior displayed in Figure 13 does however resemble what was found in [12]. The authors predict that at a critical value γ_c, the curve will change behavior going from being monotonically decreasing to having a varying sign in its derivative. This effect was not identified clearly, at least not for the expected value of $\gamma_c = \frac{1}{2}$. As one can see in Figure 13, $\gamma = 0$ displays the overall trend of the monotonically decreasing function, however with some noise. It is suspected that with more realizations of these systems, this noise could be reduced and one might more easily identify γ_c. For individual plots of all the results, see

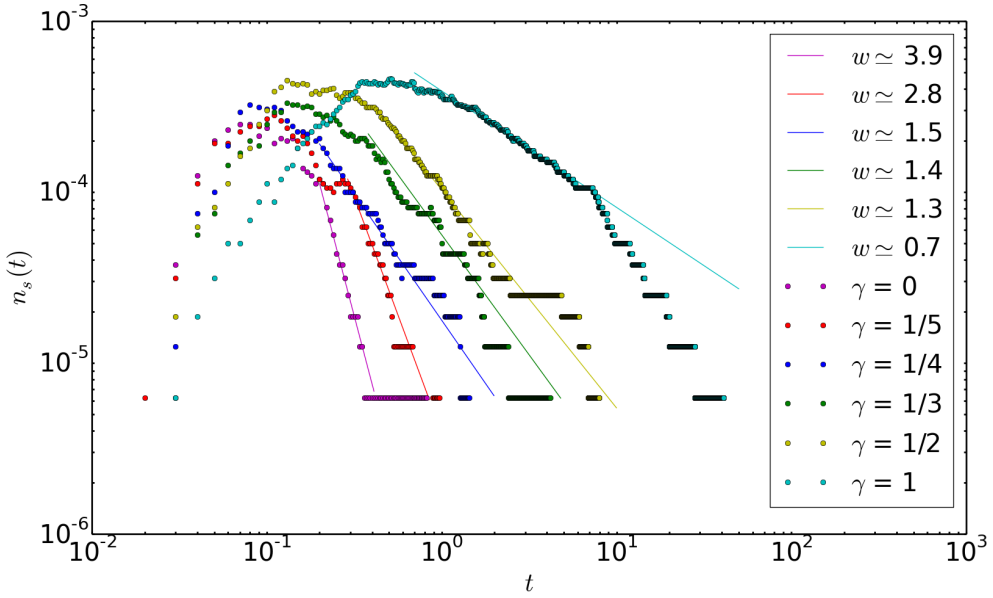


Figure 11: Relation between diffusivity (5) and time evolution of the cluster size distribution (4). Simulations were done using $\rho = 0.0625$ on a $L = 400$ lattice with a fixed $s = 10$. The dots represent the results from the simulations, while the solid lines are the least square fit approximation to $n_s(t) \sim t^{-w}$.

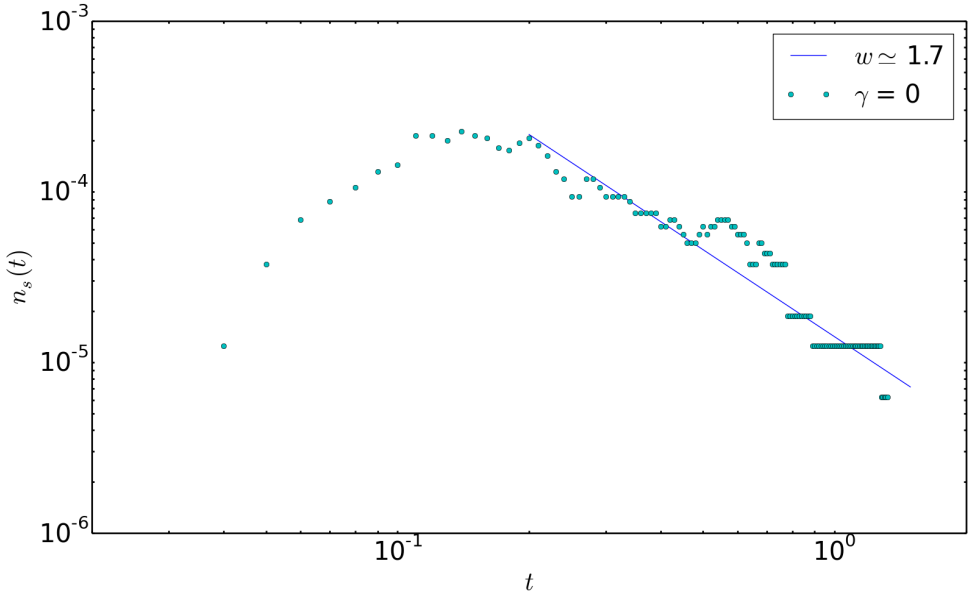


Figure 12: Relation between diffusivity (5) and time evolution of the cluster size distribution (4). Simulations were done using $\rho = 0.05$ on a $L = 400$ lattice with a fixed $s = 10$. The dots represent the results from the simulations, while the solid line is a curve of slope $w = 1.7$ (not a least square fit).

Figure 18 in the A Appendix.

Another difference when comparing the results is that for $\gamma \in [0, 1]$, the curves

seems to be broader compared to what was found in [12]. This is suspected to be due to the weakness described in 6.1. The weakness should be possible to eliminate, given some time. A possible solution would be to give the cluster a new direction to move in after a collision occurs, and calculate the new step length weighted by the amount of particles the cluster originally crashed into.

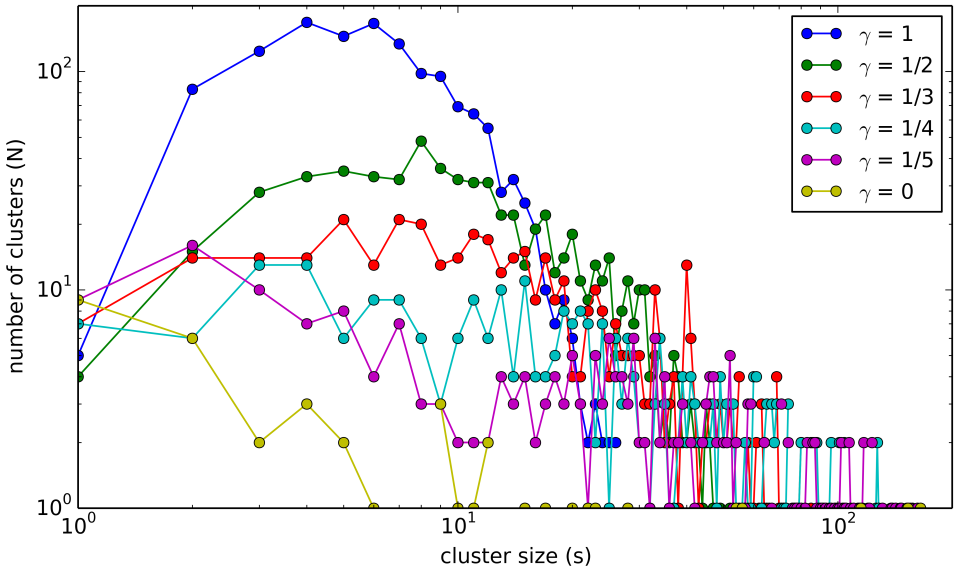


Figure 13: Relation between diffusivity (5) and cluster size distribution (4). Simulations were done using $\rho = 0.0625$ on a $L = 400$ lattice with a fixed $t = 0.6$.

7 Deposition model

In this section, a crude model for deposition of the clusters is presented. This is done in order to get some insight into the mechanisms driving the deposition of black carbon aerosols, but is not explored in great detail.

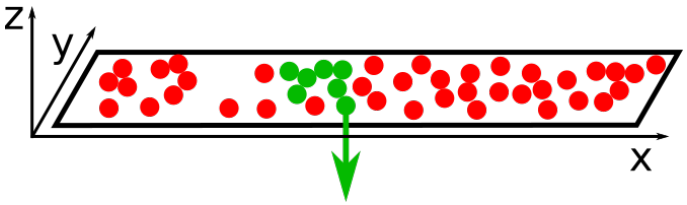
7.1 Method

The model is based on the idea that the clusters need to obtain a critical mass before they can have a non-negligible fall velocity. In this crude model, only identical particles are modeled. The only way the clusters these particles form may obtain a non-negligible fall velocity is if the cluster exceed some threshold size S_{crit}, meaning they are massive enough to fall down. Any cluster exceeding this limit is removed from the domain (i.e. it falls out), and single particles are introduced at random locations to maintain conservation of mass. The average cluster flux (number of clusters falling out) $\Phi_c = \frac{\#\text{clusters out}}{t}$ out of the plane is then analyzed, and investigated for different values of S_{crit} and γ to investigate the mass-diffusivity relation (equation (5)). All diffusion takes place in the plane (two dimensions), so that the clusters falling out will fall along a third dimension. An illustration is shown in Figure 14. It is expected that the clusters will take some time to become large enough to reach S_{crit}, but that the system will after some time reach a steady state where particles in and clusters out stabilize Φ_c.

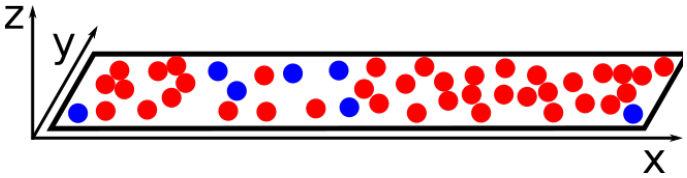
It is assumed that Φ_c will be closely related to the deposition rate, and is of more interest than the particle flux (number of particles falling out) Φ_p. This is because the mean size of the clusters falling out, $\bar{S_\Phi}$, is not expected to exceed S_{crit} by a large margin for small values of S_{crit}. Additionally, one assumes that the number of clusters deposited are of greater interest than the number of particles in each cluster when looking at meteorological effect (as suggested by section 3). The simulations for the deposition rates were performed using $N = 3000$ particles on a $L = 200$ domain, giving a density of $\rho = 0.075$. Like in section 6.1, periodic boundary conditions were used. The simulations were carried out for 3000 iterations of $\Delta t = 0.01$.

7.2 Results and discussion

As discussed in the previous section, the cluster flux Φ_c out of the plane is assumed to be of greater interest than particle flux Φ_p, under the assumption that the mean size of the clusters removed $\bar{S_\Phi} \approx S_{\text{crit}}$. Simulations were therefore run to prove this, for $\gamma = \frac{1}{2}$ and $\gamma = 1$, and the results can be found in Table 1. These results indicate that the assumption holds for small values of S_{crit}. This is also to be expected, as there will predominantly be particle-cluster collisions for small S_{crit}. The collisions will change towards cluster-cluster collision when demanding a larger S_{crit}. The latter explains the large deviation for $S_{\text{crit}} = 50$, $\gamma = 1$ in Table 1b.



(a) The (green) cluster exceeding the threshold size S_{crit} will be heavy enough to fall out of the plane. The remaining particles are unaffected.



(b) After the cluster has been removed, new (blue) particles are randomly distributed in the plane to conserve mass.

Figure 14: Illustration of how the deposition model works. When a cluster exceeds the threshold S_{crit} (in this example $S_{\text{crit}} = 6$), the cluster will fall down, and thus exit the plane. This is illustrated in 14a. To conserve mass, new particles are introduced as shown in Figure 14b. All of this happens during a single time iteration.

Table 1: The computed mean size of the clusters $\bar{S_\Phi}$ falling out of the plane, given a threshold size of S_{crit} for falling down and a relation between diffusivity and cluster size s (see equation (5))

(a) Results for $\gamma = \frac{1}{2}$.

S_{crit}	$\bar{S_\Phi}$	Deviation
10	11.11	11.11%
20	22.32	11.60%
30	33.57	11.90%
40	45.23	13.07%
50	56.63	13.26%

(b) Results for $\gamma = 1$.

S_{crit}	$\bar{S_\Phi}$	Deviation
10	10.75	7.50%
20	21.46	7.30%
30	32.68	8.93%
40	42.00	5.00%
50	70.00	40.00%

The threshold S_{crit} was investigated in the range $[10, 50]$ with steps of $\Delta S_{\text{crit}} = 10$ for 3000 time steps $\Delta t = 0.01$. Additionally, the effect of the mass-diffusivity relation (5) was studied, using steps of $\Delta\gamma = 0.1$. Results are given per unit time, i. e. clusters per $100\Delta t$. The results can be found in Figures 15 and 16.

Inspection of Figure 15 suggests that $\Phi_c(S_{\text{crit}}, \gamma)$ decays as as a power law with S_{crit}, so one makes the ansatz that $\Phi_c \sim (S_{\text{crit}})^\omega$. This is done under the assumption that the system will reach a steady state where Φ_c stabilizes. The values of ω was determined by doing a least square fit on the curves in 15, and the results can be found in Table 2.

Another interesting result is shown in Figure 16 where there results of the simulations with a strong mass-diffusivity relations are presented. Here one sees a

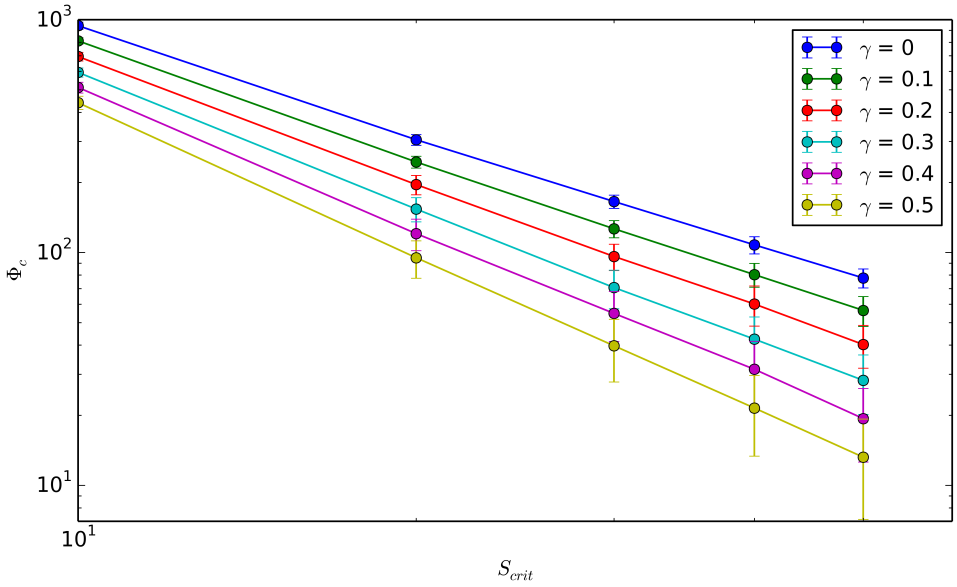


Figure 15: Φ_c as a function of S_{crit}, evaluated for $\gamma \in [0, 0.5]$ with $\Delta\gamma = 0.1$. One standard deviation is included as error bars. The simulation was done using $N = 3000$ particles on a $L = 200$ lattice, with 3000 iterations.

deviation from the power law decay suggested earlier, especially for large γ. This is however according to what can be expected, as this represents simulations where the system does not have time to reach a steady state. Even in the cases where the system does eventually reach this state, the build-up phase will dominate the steady state contribution when calculating Φ_c, and the curve deviates from the expected behavior. It it thus apparent that running 3000 iterations is not enough to analyze systems characterized by $\gamma < 0.5$.

For completeness, the dependence on γ was plotted in Figure 17. A curve fit was done to determine the decay of Φ_c, and the results can be found in Table 3 in the A Appendix.

It has been stated that this is a crude model, so some suggestions as to how this

Table 2: Curve-fitted slopes of the data in Figure 15.

γ	Slope (ω)
0	-1.6
0.1	-1.7
0.2	-1.8
0.3	-1.9
0.4	-2.1
0.5	-2.2

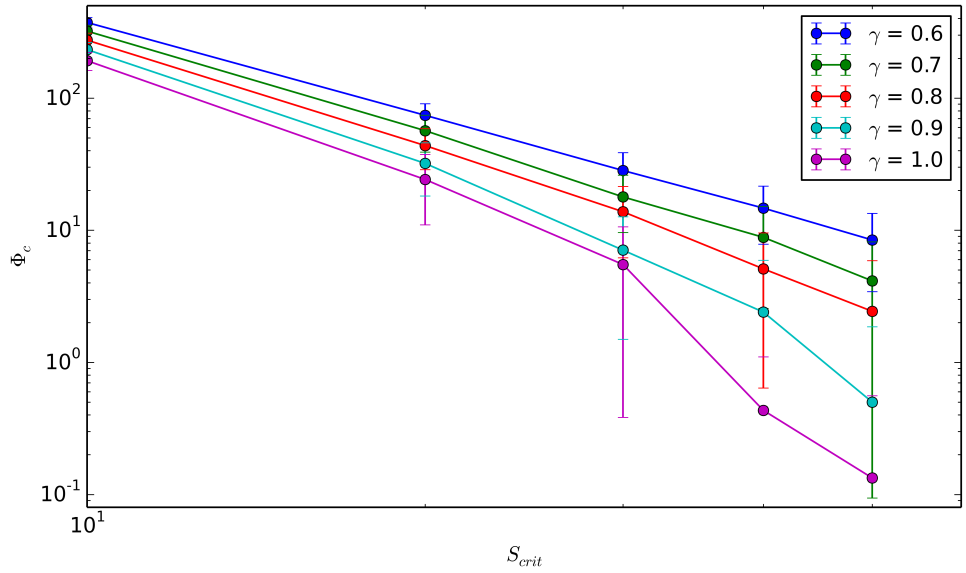


Figure 16: Φ_c as a function of S_{crit}, evaluated for $\gamma \in [0.6, 1]$ with $\Delta\gamma = 0.1$. One standard deviation is included as error bars, except for the simulations where the system is far from reaching a steady state. The simulation was done using $N = 3000$ particles on a $L = 200$ lattice, with 3000 iterations. Note that the system does not reach its steady state for the larger values of γ, causing the deviation from a power-law decay.

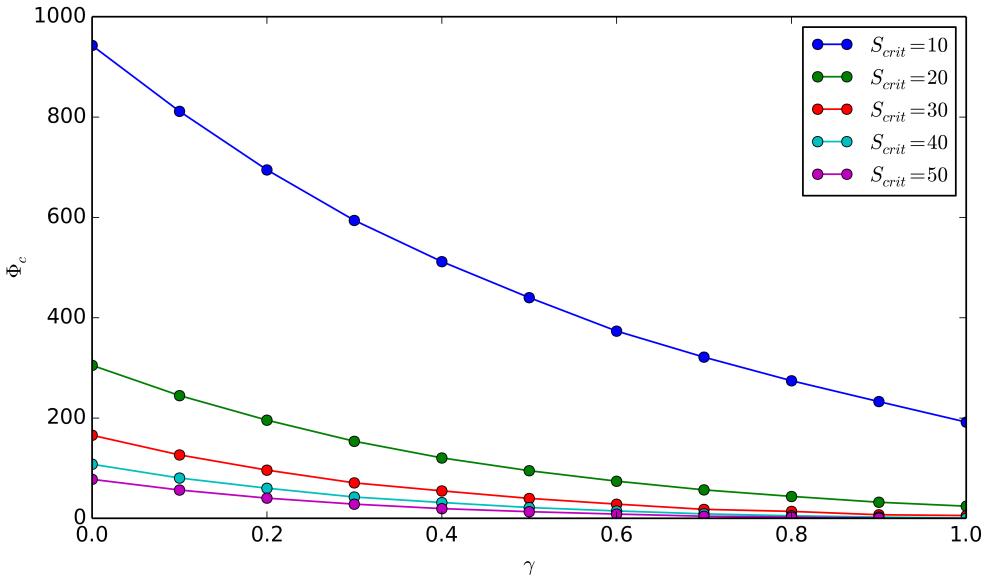


Figure 17: Φ_c as a function of γ, evaluated for $S_{\text{crit}} \in [10, 50]$, with $\Delta S_{\text{crit}} = 10$. The simulation was done using $N = 3000$ particles on a $L = 200$ lattice, with 3000 iterations.

can be improved are made. Previous studies suggests that the clusters formed exclusively by black carbon aerosols will never obtain non-negligible fall velocity. Figure 10.1 in Seinfeld [19] suggests that the clusters formed will never exceed $\sim$ 1 µm. The single BC particles are expected to have a diameter $\sim$ 10 nm. Flanagan

and Seinfeld [7] suggests that the fall velocity of 1 µm aerosols will be $\sim 3.5 \times 10^{-3}$ cm/s (see the top Table on page 300), which is negligible compared to turbulent flows. The results obtained in [19] are based on the theory by Fuchs [20]. Taking this into account, it seems evident that the model presented here is not well suited for completely describing the proposed problem. A possible solution to this may be that clusters interact with other primary particles, which may be heavy enough to drive the deposition. Generally, this may be precipitation, but since one in this case is interested in desert areas, others solids may be investigated.

It seems reasonable to assume that in desert areas there would be a low density of liquids in the atmosphere, and a high density of mineral dust. These mineral dust particles are heavy relative to the BC aggregates ($M_{\text{mineral dust}} >> M_{\text{BC cluster}} >> M_{\text{single particle}}$), since the aggregation of large clusters is a slow process [19], [21]. A natural way to improve the model presented here is to instead let the deposition rate be governed by collision with mineral dust particles, instead of allowing the clusters to become heavy enough to fall down on their own. This is also the reason why the method presented in section 6 allows for particles of varying size.

8 Conclusion

During the work done in this specialization project, the DLA algorithm for irreversible growth has been investigated, both on and off lattice. This was done in order to develop a non-Monte Carlo simulation algorithm for the aggregation of moving clusters (DLCA). The method used in this work differs from the traditional ways of simulating the process, and the differences have been investigated. The approach taken in this work has what appears to be a non-negligible loss of net movement, which is the reason why the results presented differ somewhat from previous works. However, the overall results does agree very well. A crude model for deposition of agglomerates has been proposed, and the impact the relation between mass and diffusivity has on this model has been investigated. Some suggestions to how this model may be improved has been discussed.

9 References

[1] J. M. Haywood and K. P. Shine. The effect of anthropogenic sulfate and soot aerosol on the clear sky planetary radiation budget. *Geophysical Research Letters*, 22(5):603–606, 1995.

[2] D. Koch and A. D. Del Genio. Black carbon semi-direct effects on cloud cover: review and synthesis. *Atmospheric Chemistry and Physics*, 10(16):7685–7696, 2010.

[3] T. C Bond, S. J Doherty, D. W Fahey, P. M Forster, T. Berntsen, B. J DeAngelo, and et al. Bounding the role of black carbon in the climate system: A scientific assessment. *Journal of Geophysical Research: Atmospheres*, 118(11):5380– 5552, 2013.

[4] J. Hansen and L. Nazarenko. Soot climate forcing via snow and ice albedos. *Proceedings of the National Academy of Sciences of the United States of America*, 101(2):423–428, 2004.

[5] M. G. Flanner, C. S. Zender, P. G. Hess, N. M. Mahowald, T. H. Painter, and V. Ramanathan. Springtime warming and reduced snow cover from carbonaceous particles. *Atmospheric Chemistry and Physics*, 9(7):2481 – 2497, 2009.

[6] M. Sand, T. K. Berntsen, K. von Salzen, M. G. Flanner, J. Langner, and D. G. Victor. Response of arctic temperature to changes in emissions of short-lived climate forcers. *Nature Clim. Change*, advance online publication, Nov 2015.

[7] Richard C Flagan and John H Seinfeld. Fundamentals of air pollution engineering. 1988.

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

[9] M. 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.

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

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

[12] P. Meakin, T. Vicsek, and F Family. Dynamic cluster-size distribution in cluster-cluster aggregation: Effects of cluster diffusivity. *Phys. Rev. B*, 31:564–569, Jan 1985.

[13] S. Berntsen. Modeling agglomeration of aerosols. `https://github.com/berntsim/Project/tree/master/Project/Code`, 2015.

[14] A. Einstein. Investigations on the theory of the brownian movement. *Ann. der Physik*, 1905.

[15] A.L. Barabási and H.E. Stanley. *Fractal Concepts in Surface Growth*. Cambridge University Press, 1995.

[16] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Advanced book program. Westview Press, 1994.

[17] Paul Meakin. Diffusion-controlled cluster formation in 2⁻6-dimensional space. *Phys. Rev. A*, 27:1495–1507, Mar 1983.

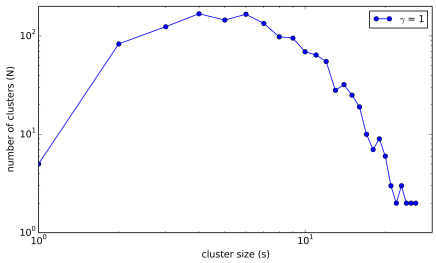
[18] K. R. Kuijpers, L. de Martín, and J. R. van Ommen. Optimizing off-lattice diffusion-limited aggregation. *Computer Physics Communications*, 185(3):841 – 846, 2014.

[19] John H Seinfeld. *Atmospheric chemistry and physics of air pollution*. Wiley, New York, 1986.

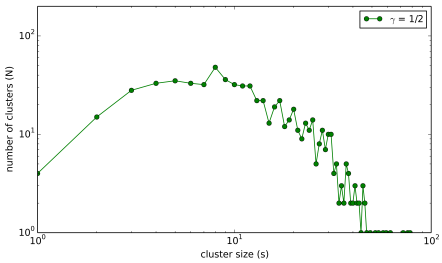
[20] N. A. Fuchs, R. E. Daisley, Marina Fuchs, C. N. Davies, and M. E. Straumanis. The mechanics of aerosols. *Physics Today*, 18(4):73–73, 1965.

[21] 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.

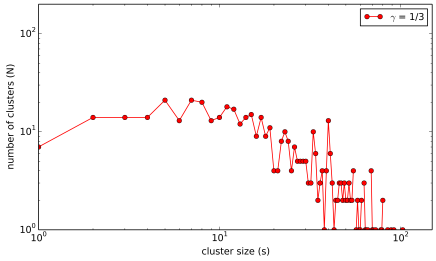
A Appendix



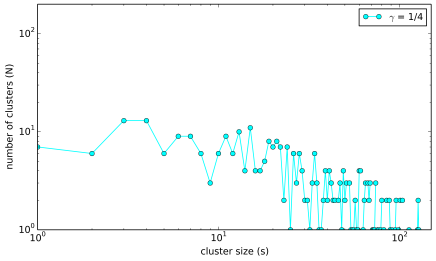
(a) $\gamma = 1$



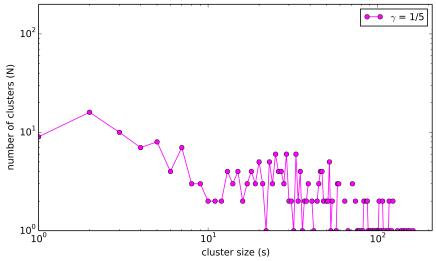
(b) $\gamma = \frac{1}{2}$



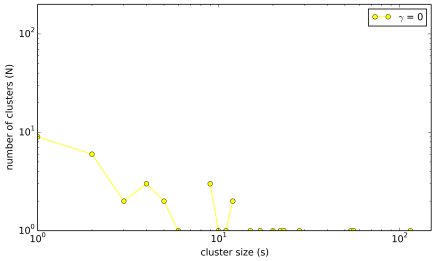
(c) $\gamma = \frac{1}{3}$



(d) $\gamma = \frac{1}{4}$



(e) $\gamma = \frac{1}{5}$



(f) $\gamma = 0$

Figure 18: Similar to Figure 13, but now with individual plots for all simulations. Be aware that the scales of the axes may vary.

Table 3: Curve-fitted slopes of the data in Figure 17. It is assumed that the curves can be represented by $\Phi_c \sim e^{c\gamma}$

S_{crit}	Slope (c)
0.1	-1.5
0.2	-2.2
0.3	-2.6
0.4	-2.8
0.5	-3.0