University of Oxford

# Effects of Particle Attraction and Size on the Stability of Diffusion Limited Aggregation

*Candidate Number:*
213032

*Course:*
MMath Mathematics

BSP: Structured Project

Hilary Term
2015

# Contents

# 1 Introduction

Proposed by Witten and Sander in 1981 [18], Diffusion Limited Aggregation (DLA) is a model for a range of natural phenomena that have fractal growth. The model begins with a static seed in the centre and another particle is launched at a large distance from the seed. This new particle diffuses through the space by Brownian motion until it is adjacent to the seed, at which point it attaches and becomes part of the cluster. Another particle is then launched from a random distant point and walks randomly until it attaches to the cluster. Particles are continually added until the model is stopped by some parameter. The model can also be simulated on-lattice with the use of random walk. Despite its relatively basic growth pattern DLA has proven difficult to model and has become one of the most studied theoretical aggregation processes, with both on and off lattice models thoroughly investigated. A good summary of investigations was written by Sander in 2000 [16]. DLA forms the basis for simulations on a wide range of topics including nanoparticle agglomeration [20], bacterial colonies [11] and electrodeposition [12]. Figure 1 displays a crystal grown by electrodeposition in zinc sulfate solution exhibiting DLA.

The optimisation and structural analysis of DLA has provided a long-lasting challenge. As the cluster size increases, the density of DLA decreases due to the creation of deep fjords inbetween long branches, and particles then spend significant amounts of time walking in these fjords before aggregating. Clusters built via DLA exhibit a type of scaling pattern whereby the radius of a cluster grows according to a power law relationship with the number of particles in the aggregate. When a drift has been applied to DLA it has been found that DLA endures a crossover to Ballistic Aggregation (BA) [13]. Motivated by this crossover of structure we investigate how varying the radii of the particles affects the stability and scaling of the cluster and whether any crossover occurs. Following on, we apply a type of drift whereby particles are attracted to one another depending on their radii and equally investigate whether DLA maintains its defining features. For the duration of these investigations we needed to maintain the optimisation of our simulation and at each stage we discuss how this is possible.

A brief synopsis of the paper content is as follows. In Section 2 we describe the model used, its implementation in MATLAB and a brief critique of modelling on-lattice is discussed. In Section 3 we begin investigating the stability of DLA by releasing particles of different radii firstly by means of a discrete distribution between two distinct radii and secondly by use of various continuous distributions. Section 4 continues these investigations by applying particle to particle attraction and combining this with cluster of varying radii with larger particles offering larger attraction.

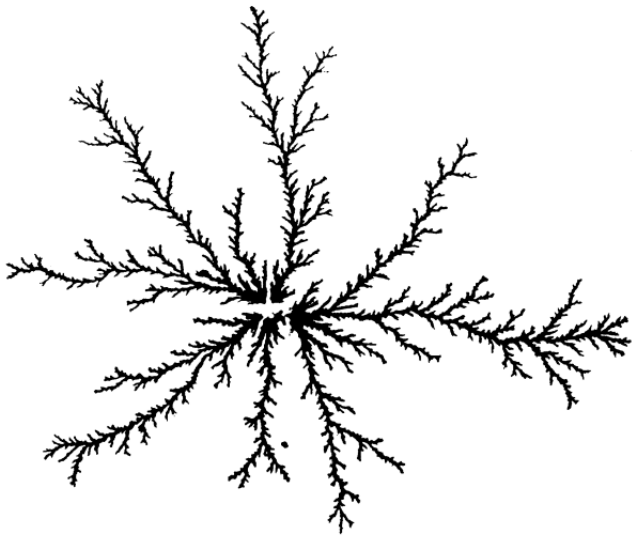


Figure 1: Crystal grown in zinc sulfate solution exhibiting DLA. The picture is from [12].

# 2 Optimising the Algorithm

## 2.1 Models and Methods

To produce large scale simulations of the model we would suffer extreme computation times if we simply adhered to the algorithm of DLA described above. Particles would spend enormous amounts of time walking before aggregating with large distances to cover before reaching the cluster. Therefore we must apply techniques to optimise the algorithm and provide an efficient way of producing clusters. In our model we do this by making use of a computer's memory as has been investigated previously [9]. The aggregation modelling presents us with four main issues that need addressing: simulating the launch of a walker from infinity, the walker spending a large amount of time walking far away from the cluster, the walker spending a large amount of time in deep fjords in between branches of the cluster and identifying when the walker will attach to the aggregate. We explain how each of these points has been addressed in our model.

If we draw any circle in space a walker performing Brownian motion from infinity will hit the boundary of the circle with a uniform distribution. We enclose our cluster in a circle slightly larger than the cluster of radius $R_{launch}$ and launch our new walkers from a uniformly random point on this circle. In our model we let $R_{launch} = R_{max} + 5L_{min}$ where $R_{max}$ is defined to be the largest distance from the centre of the lattice of any particle in the cluster and $L_{min}$ is the smallest step size that can taken. Even if we start the walker on this launching circle it may well wander off in the opposite direction and could spend significant time far away from the cluster. A walker outside the launching circle will hit the launching circle with a certain probability distribution depending on where the walker is in space [15]. We can find this probability density function via a Möbius Transformation and at any time outside the launching radius bring the walker back in one step. If the walker is at a point with distance

$a$ from the seed, with $a > R_{launch}$, then, setting $\theta = 0$ to be the angle from the seed to the walker, the walker hits the launching radius with probability density function

$$f(\theta) = \frac{1}{2\pi} \frac{a^2 - R_{launch}^2}{a^2 - 2aR_{launch}\cos\theta + R_{launch}^2}.$$

This calculation can be quite costly if it were triggered often, for example every time the walker went outside the launching radius. Therefore when using this method we need to find an optimal distance so that this calculation is performed rarely and only when needed. However, if we let $a = nR_{launch}$, for $n \in \mathbb{N}$, we note that as the distance from the centre of the walker increases, the probability density distribution approaches the uniform distribution on the launching circle, since then

$$f(\theta) = \frac{1}{2\pi} \frac{n^2 - 1}{n^2 - 2n\cos\theta + 1} \longrightarrow \frac{1}{2\pi}, \quad \text{as } n \longrightarrow \infty.$$

This observation can lead us to use a different approach by which we abandon the time costly calculations involved in bringing back the particle in one step and employ a killing radius, $R_{kill}$, whereby if a walker walks further than this distance from the centre it is removed and relaunched from the launching radius uniformly. The walking outside of the launching radius here is not a problem since we can take large steps and so even though we may take some steps outside the launching radius this is more efficient than performing calculations to relaunch the particle from the launching radius. Large $n$ increase computation times however we seek $n$ large enough so that it provides a good approximation of our uniform distribution. After investigation it was found that for our model the optimal set up was to utilise a killing radius of $R_{kill} = 5R_{max}$. For $n = 5$ the value is large enough to approximate well a uniform distribution but still provide good simulation speeds. This can be seen by seen by comparing the difference in density between the point of highest probability, $\theta = 0$, and the point of lowest, $\theta = \pi$. For $n = 5$ we gain a difference of $5/12\pi$ while increasing our parameter to $n = 10$ only roughly halves this difference not justifying its increased simulation times.

We now turn our attention to the issue of our particle spending increasingly large amounts of time inbetween branches of the cluster, the so-called deep fjords. In a cluster of $10^6$ particles these deep fjords could be many thousands of times the radii of particles and walking with small steps would be ineffective. As above we employ a tactic of being able to take large steps when far away from the aggregate. For this to be possible at any point in space the walker must be able to find out approximately how far it is from the aggregate and in our model we make use of large computer memory to store matrices of this information. We must define a maximum step size, $L_{max}$, which is the largest step a particle can take. Ideally we want $L_{max}$ as large as possible but, as we will see below, this increases computation time and so an equilibrium must be found. Once we have $L_{max}$ we can define the Vicinity Matrix which we will calculate at the beginning and will remain a constant throughout the modelling. This matrix will be a $2L_{max} + 1$ square matrix and each cell will contain the rounded down distances between the cell and the middle cell of the matrix, where if this distance is greater than $L_{max}$ we let the cell entry be $L_{max}$. An example of the Vicinity Matrix is shown in Figure 2 where we have set $L_{max} = 8$.

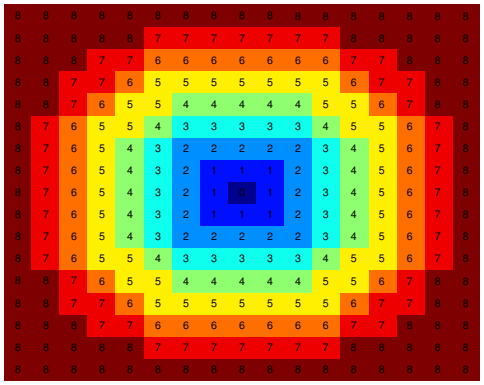
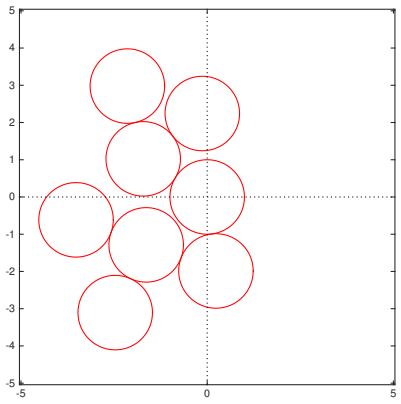


Figure 2: Vicinity Matrix with $L_{max} = 8$.

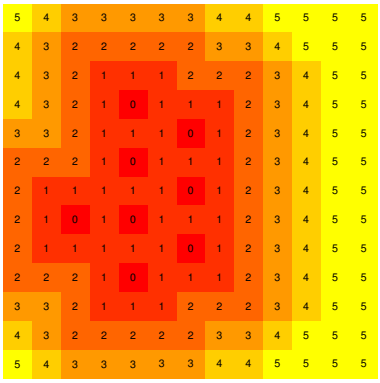
We set up three additional matrices: the Stuck Particles Matrix which stores the exact coordinates of aggregated particles, the Cluster Matrix which will store an on-lattice version of the cluster with the cell value equal to the particle label, i.e. the time the particle aggregated. In the off-lattice case to obtain the on-lattice version we round the aggregated particle's real coordinates to the nearest integers. To ensure that at most one particle refers to each cell we set the radii of the particles to at least one so no two particles can be rounded to the same cell. The final matrix, the Distance Matrix, will contain the rounded down distances between each cell and the nearest occupied cell according to our Cluster Matrix with occupied cells taking the value 0. If a cell is further than $L_{max}$ from an occupied cell then this cell will take the value of $L_{max}$. Figure 3 gives an example of how the Cluster Matrix and Distance Matrix might look near the beginning of a simulation.

Every time a particle is aggregated we add the particle's coordinates to the Stuck Particles Matrix, update the relevant cell in the Cluster Matrix and recalculate some of the cells in the Distance Matrix. The cells we might have to recalculate are contained in a square centred on the new stuck particle of side lengths $2L_{max} + 1$. To update these cells we compare the entry in each cell of the sub matrix to the corresponding entry in our Vicinity Matrix and take the minimum out of these two values. This method is the crucial optimisation step since at every step we can now instantly see roughly how far we are from the cluster by checking the Distance Matrix. As mentioned above, maximising $L_{max}$ would seem to be the sensible priority as then larger steps could be taken, however after each aggregation we must recalculate a $2L_{max} + 1$ square matrix which is costly. We investigated various values for $L_{max}$ and found that the optimal values for clusters with $10^5$ particles is $L_{max} = 82$ and for clusters with $10^4$ particles $L_{max} = 27$.

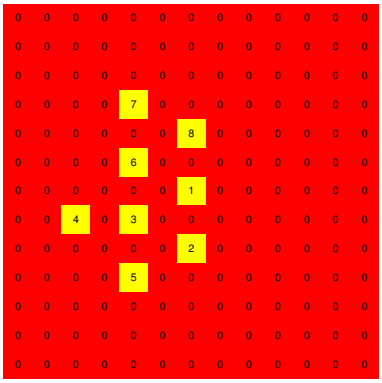
Finally our model should be able to tell when a walker is aggregated. In the on-lattice model this is relatively simple, since a walker at a certain cell is aggregated if any of the four adjacent cells contain a stuck particle. This is an easy check and does not cost much in terms of efficiency and is where the on-lattice model gains its speed advantage over the off-lattice case. Unfortunately for the off-lattice case it is not so simple. By the Markov property of Brownian motion if we draw any circle around the walker which does not intersect with the aggregate then our walker will hit the boundary of this circle uniformly. We thus draw circles around the particle as large as we can without hitting the aggregate and move the walker




(a) Cluster in space




(b) Distance Matrix




(c) Cluster Matrix

Figure 3: The beginning of a simulation in our various matrices with $L_{max} = 5$.

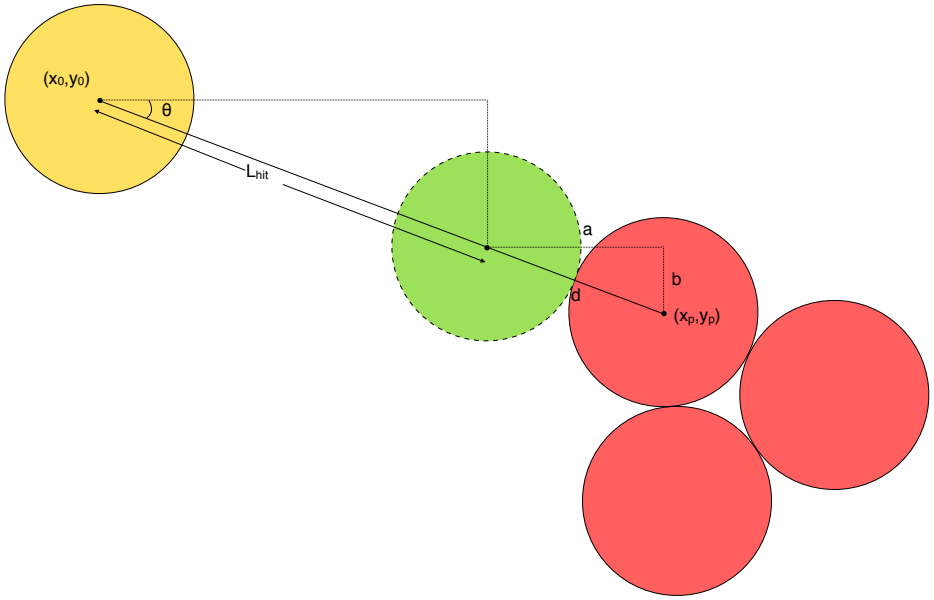



Figure 4: Calculating the hitting distance from a particle. The yellow particle represents the walker, the red particles aggregated and the green particle is where the walker should be aggregated to if $L_{hit}$ satisfies $0 < L_{hit} \leq L_{min}$.

to the boundary of this circle. If we continued in this manner as the particle gets closer and closer we draw smaller and smaller circles and the particle will never hit the aggregate since it hits any point on the circle with probability zero. Therefore as well as a maximum step size we must equally define a minimum step size, $L_{min}$, where we only aggregate if we are close enough to be able to hit the aggregate in less than this step size. We cannot have $L_{min}$ too large as otherwise we would deviate from Brownian motion but having a small value increases computation time thus in our model we use $L_{min} = r/5$ where $r$ is the smallest radius of a particle. We only need to check for aggregation when the walker is in a cell where the corresponding value in the Distance Matrix is low enough so that we could hit the aggregate at the next step. This hitting distance will be equal to twice the particle radii add the minimum step size and add one for rounding errors, so, $D_{hit} = 2r + L_{min} + 1$ where $r$ is the radius of the particles. Looking at the submatrix of sufficiently close entries around the walker we can extract from the Cluster Matrix the relevant particle labels and then by choosing a random angle to walk uniformly we can calculate the distance needed to collide with each of these particles. If the distance is less than $L_{min}$ then we walk this distance and attach to the aggregate.

We can calculate this distance through some algebra and geometry. Figure 4 displays the scenario where $\theta$ is our random angle chosen and $L_{hit}$ is the distance to aggregate. From this

we see

$$\begin{aligned} a &= x_p - (x_0 + L_{hit}\cos\theta), \\ b &= y_p - (y_0 + L_{hit}\sin\theta), \\ d^2 &= a^2 + b^2. \end{aligned}$$

Since $d$ is simply the distance between the two particle centres, $d$, is the sum of the two particle radii which is known and we can rearrange these equations to give us the quadratic equation

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

where

$$\begin{aligned} A &= 1, \\ B &= 2(\cos\theta(x_0 - x_p) + \sin\theta(y_0 - y_p)), \\ C &= (x_p - x_0)^2 + (y_p - y_0)^2 - d^2. \end{aligned}$$

Solving this quadratic equation, if there exists a solution then we take the minimum solution that satisfies $0 < L_{hit} \leq L_{min}$ . Otherwise if the solutions are outside this range or do not exist then the particle will not aggregate in a step of size $L_{min}$. We need to check for each particle sufficiently close if there is to be collision and in the case of collision with multiple particles in the aggregate we choose the smallest distance to eliminate the chance of overlapping.

This method was chosen as it can be easily adapted to perform the model when the particle radii are not uniform, as long as the smallest radius is of unit size then the model will hold. We acknowledge that methods such as the Brady-Ball method [3] may provide slightly more efficient means of constructing models however since investigating stability, not optimisation, is the purpose of this paper we happily proceed with our above model. An off-lattice cluster built with $100,00$ particles is shown in Figure 5 and was built in 35 minutes.

## 2.2 On-Lattice Anisotropy

In the on-lattice case when following the model described above after each step we round ourselves to the nearest integer cell. When close enough to the aggregate then random walk on the lattice is used. As mentioned above since the aggregation check on a lattice is significantly easier then we are able to simulate large scale models of DLA in sensible time. Figure 6 shows an on-lattice model with $10^5$ particles which was grown in under 2 minutes. The colouring represents the arrival time of the particle to the cluster and from this Figure we can see one of the key properties of DLA, that particles mostly attach to large branches leading to the growth of deep fjords.

In the on-lattice models particles are approximated by square cells and thus a walker cannot aggregate to anywhere around the aggregated particle as in the off-lattice case. It is restricted to attaching in one of the four axial directions and as a result of this, the on-lattice model

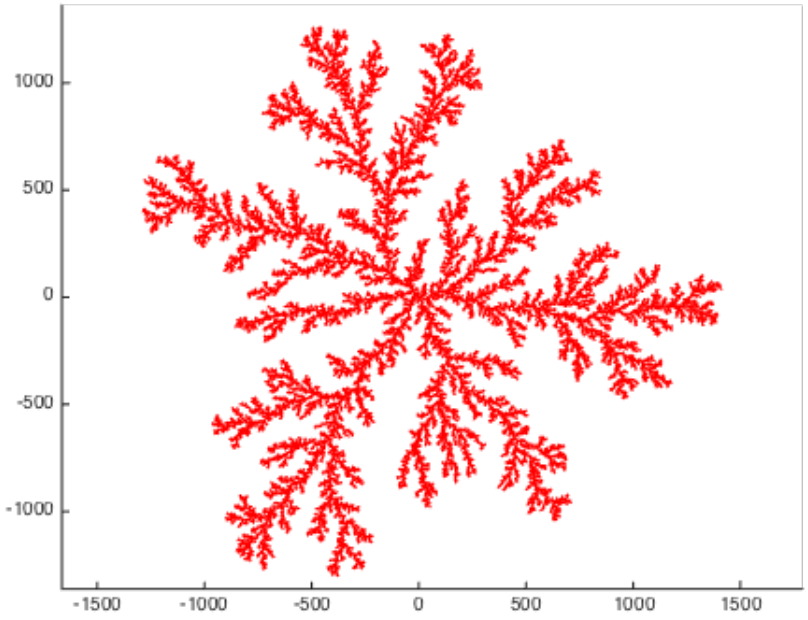



Figure 5: Off-lattice cluster of $100,000$ particles.

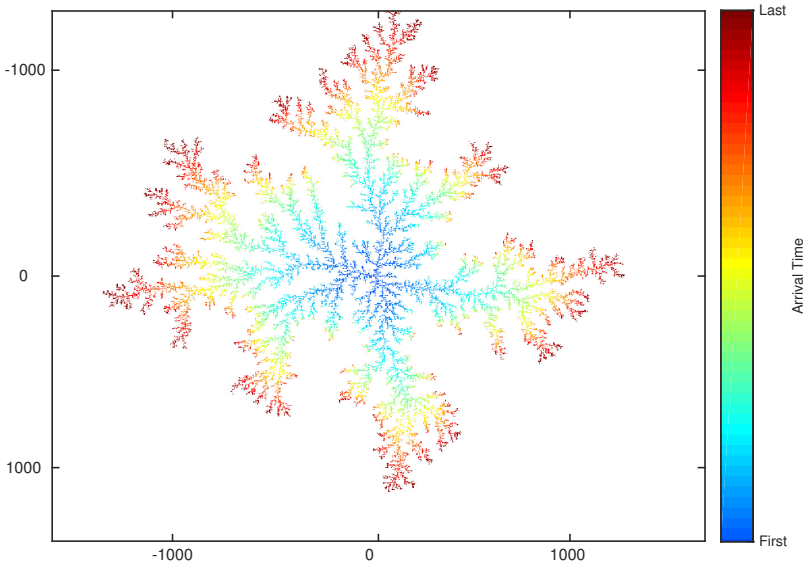



Figure 6: On-lattice cluster with $10^5$ particles where colouring of the cluster relates to the arrival time of the particle.

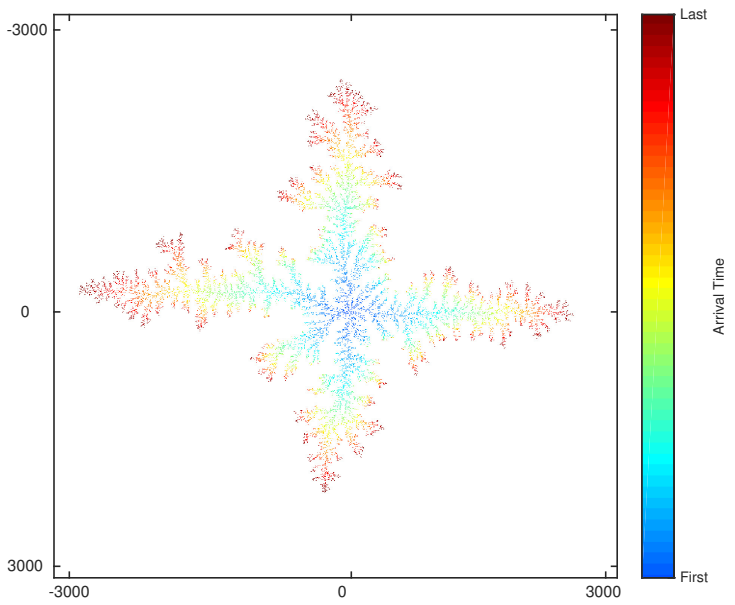



Figure 7: On-lattice cluster with $10^6$ particles displaying clear anisotropy.

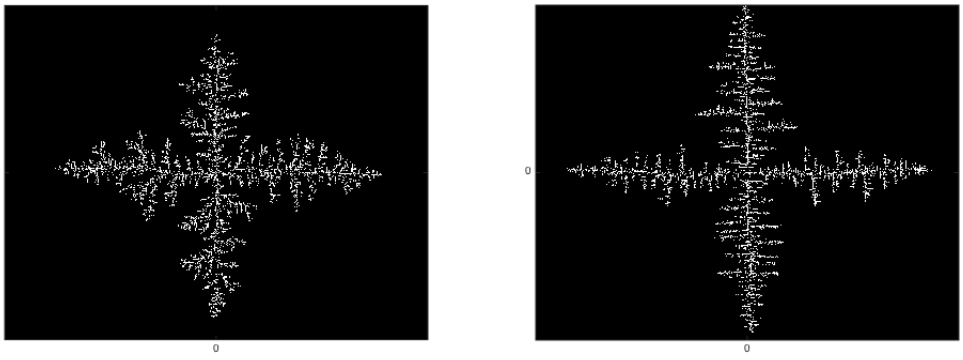
produces anisotropic results with preferred growth in the axial directions. We can demonstrate the anisotropic behaviour of the on-lattice model through various methods. Our first and perhaps most primitive method is by producing large-scale cluster of over $10^6$ particles as can be seen in Figure 7 where axial preferential growth is evident. Here we can clearly see growth in the axial directions significantly larger than in any other directions.

We can also apply a type of probabilistic exaggeration to the model whereby we require walkers to visit a site at least $M$ times before we aggregate it. Then if anisotropy is present the preferred growth of directions will be magnified while improbable growth directions will become even more unlikely. With this exaggeration applied it does not take large cluster sizes to display anisotropy and Figure 8 displays clusters of $50,000$ particles with various values for $M$. In addition to above we can apply a quantitive measure of the anisotropy in terms of the fourth circular harmonic [3]

$$\langle \cos 4\theta \rangle = \frac{1}{N} \sum_{i=1}^{N} \cos 4\theta_i,$$

where the summation is over the aggregated particles and $\theta_i$ refers to the angle of the aggregated particle relative to the origin. It is known [2] that for fractals with axial anisotropy then $\langle \cos 4\theta \rangle > 0$, for fractals with diagonal anisotropy $\langle \cos 4\theta \rangle < 0$ and we have equality for isotropic fractals. Table 1 displays the average values of the fourth circular harmonic over 100 iterates for various cluster sizes. Due to its computational time requirements, the cluster with $10^6$ particles was only averaged over 10 iterates. Other methods of displaying anisotropy exist, however with this third dataset we conclude that we have adequately displayed the anisotropy of on-lattice DLA.

As a result of the clear anisotropy displayed by the on-lattice model, despite the relative



(a) $M = 2$ (b) $M = 4$

(c) $M = 8$ (d) $M = 16$

Figure 8: On-lattice cluster with $50,000$ particles where we require a site to be visited $M$ times before the particle is aggregated.

| Number of Particles | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
|---|---|---|---|---|
| $\langle \cos 4\theta \rangle$ | 0.0796 | 0.1243 | 0.2395 | 0.4321 |

Table 1: Average fourth circular harmonic for various cluster sizes.

speed at which you can produce large DLA clusters, we proceed by my making use of the off-lattice model. Investigations have been made into reducing the anisotropy of DLA [4, 19], however these studies have failed to find a definitive answer leaving delaying the anisotropy to larger clusters or rotating the direction of the anisotropy as the two best current solutions to this problem.

# 3 Particle Radii Affecting Stability

We seek to characters the shape and size of fractals in very much the same way as in geometry. Where a line can be given a definite length and a square a definite area, we want to be able to apply some quantitive measure on fractals. This is where we can define the fractal dimension. A line has dimension one and a square dimension two, however the classic example gives the dimensions of the UK coastline to be roughly 1.2 since it exhibits properties of both one and two dimensions. Various definitions exist [5] with perhaps the most common being the box-counting dimension, where we let $N(r)$ denote the number of squares of side length $r$ needed to cover the fractal. Then if a fractal obeys the relation

$$N(r) = c\left(\frac{1}{r}\right)^D,$$

where $c$ is a constant, we say that the fractal has box-counting dimension $D$. From this we can take logarithms, re-arrange and take limits to obtain that

$$D = \lim_{r\to 0} \frac{\log N(r)}{\log 1/r},$$

This definition can easily give that the dimension of the von Koch curve is log4/log3 or that the dimension of the Sierpinski triangle is log3/log2. However, this cannot be used for DLA since when $r$ gets smaller than the smallest radius of a particle in the cluster the DLA does not reveal any additional structure and we cannot compute the dimension from taking continually smaller boxes. This is due to the fact that DLA is not strictly a fractal but a pre-fractal in that it only exhibits fractal properties up to a certain scale. Instead we make use of the mass dimension, where we extend the notion of how the mass of an object varies as its size changes. As before if we have a line and it doubles in size then we double the mass and if we double the size of a square we quadruple its mass. We pick a point $P$ inside the fractal, near the middle, and then denote by $M(r)$ the mass of the fractal in a disc of size $r$ centred on $P$. Then similarly to the box-counting dimension if a fractal obeys

$$M(r) = cr^D,$$

for some constant $c$, then the fractal has mass dimension $D$. If each particle is the same size we can take the mass of each particle to be unit which simplifies $M(r)$ to simply be the number of particles. If particles are of different radii then by letting particles of radius one have unit mass other particles are scaled such that the mass of a particle of radius $r$ is $r^2$.

Hence the formula for the mass of the cluster simplifies to

$$M(r) = \sum_{i=1}^{N} r_i^2,$$

where $r_i$ is the radius of the $i^{th}$ particle and $N$ is the total number of particles in the cluster. By taking logarithms of our power law relation we obtain

$$\log M(r) = d \log r + \log c.$$

Recognising this as a straight line equation we can then plot $\log M(r)$ against $\log r$ for various values of $r$ to give the gradient of the linear least-squares approximation which corresponds to the fractal dimension. In our model we build an aggregate until the required number of particles is aggregated and then measure the radius of the cluster. As such, it makes sense to in fact plot $\log r$ against $\log M(r)$ and note that the fractal dimension will then be one over the gradient of the linear least-squares approximation of the points. We grow various cluster sizes and note the radius of the cluster, $R_{max}$, which is defined to be the largest distance of an aggregated particle from the centre seed. Then we can plot $\log R_{max}$ against $\log M(R_{max})$ and use a linear least-squares fit to determine the fractal dimension as above. We utilise MATLAB's `robustfit` function to plot this linear fit. For simplicity from now on we refer to $R_{max}$ as simply $R$ and $M(R)$ as $M$. It has been computationally verified [17] that for two-dimensional DLA $D = 1.71$. We build cluster sizes of size 100 up to $200,000$ and plot the log-log graph in Figure 9. Here the fractal dimension is computed to be $D = 1.71$ as expected.

In our first step away from the standard model we altered the radii of particles. When releasing a new walker, with probability $p$ the particle has radius two and radius one otherwise. Figure 10 displays such a cluster in the early stages of its growth. This alteration could possibly form models for mineral decomposition with two different minerals or colloids with two insoluble particle sizes dispersed. The natural question to ask is then will this cluster obey our power law relation and if so for what mass dimension $D$? This modification will not affect the pre-fractal nature of DLA and so it is reasonable to assume that DLA will still exhibit a scaling pattern given by the power law relation. Some particles have radius two now and this increases the hitting distance $D_{hit}$ which, in turn, increases computational time since we now have to check all particles for collision in a larger area around the walker. Namely when the walker has radius two, $D_{hit} = 4 + L_{min} + 1$ and when a walker has radius one we have now, $D_{hit} = 3 + L_{min} + 1$ since the walker could be close to a particle of radius two. This slight increase has an effect on computation time and as such clusters of up to only $100,000$ particles can be built in reasonable time.

We now investigate whether our modified DLA still obeys the same scaling principles with various values for $p$. Clearly if $p = 0$ or $p = 1$ then we return to our standard model and expect to exhibit scaling of $D = 1.71$. For numerous probability values we grow clusters of varying sizes from 100 up to $75,000$ and again plot $\log R$ against $\log M$ to calculate the fractal dimension as above. Table 2 provides values for $D$ for various values of $p$ with Figure 11 displaying the growth patterns for some of these values.

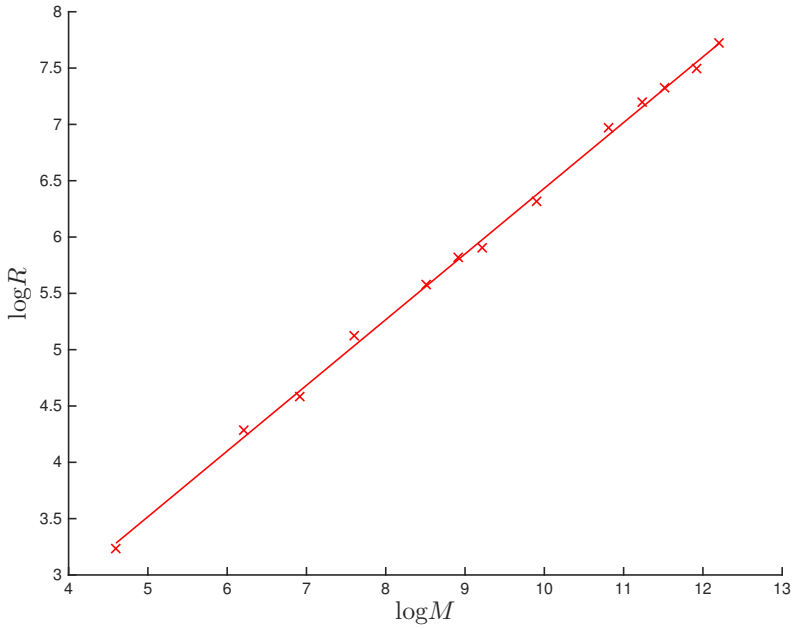



Figure 9: Plot of log$R$ against log$M$ for various cluster sizes with a linear least-squares approximation. The fractal dimension is then given by one over the gradient, here $D = 1.71$.

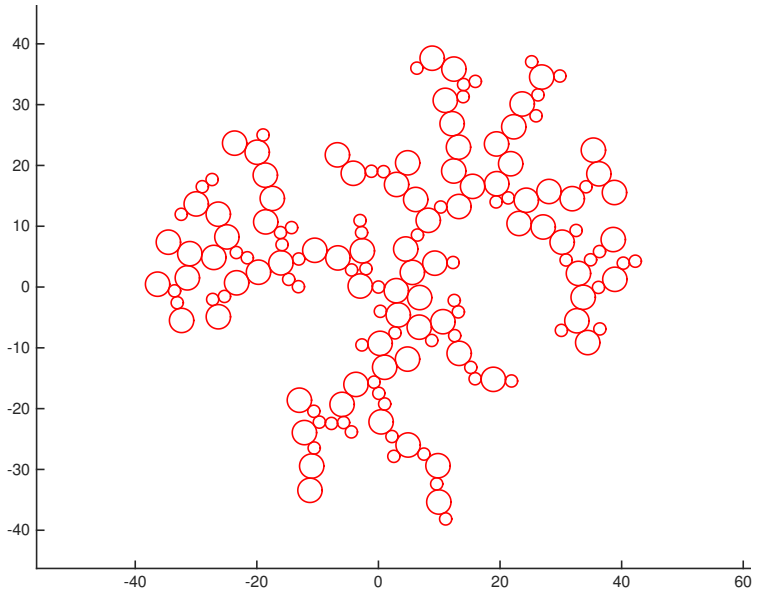



Figure 10: The early stages of cluster growth with two particles sizes of probability $p = \frac{1}{2}$ each.

| $p$ | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 |
|---|---|---|---|---|---|---|
| $D$ | 1.71 | 1.76 | 1.82 | 1.76 | 1.70 | 1.71 |

Table 2: Fractal Dimensions, $D$, for a cluster that takes particle radii two with probability $p$ and radius one otherwise.

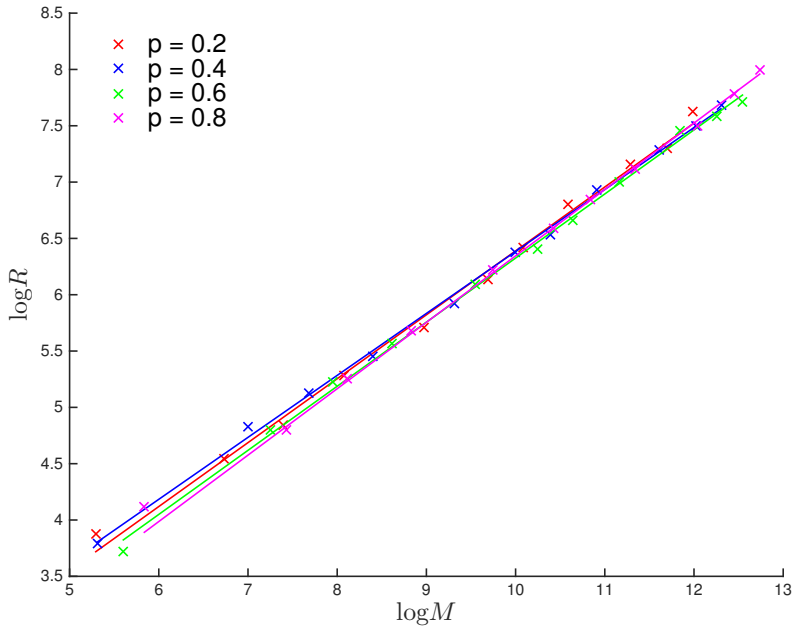



Figure 11: Growth patterns for various values of $p$ where $D$ is one over the gradient of each least-squares linear approximation.

As an aside let us comment on the accuracy of our values for $D$, not only here but throughout the duration of the paper. To calculate $D$ we plot linear best fits for eleven data points ranging from size 100 to $10^6$ and to an even lower size for difficult simulations later in the paper. We repeated numerous data points to find that for the smallest clusters our data exhibits a $\pm 0.2$ confidence interval for $\log R$ while the largest clusters give $\pm 0.01$. With these confidence intervals by looking at the largest and smallest gradients possible for our linear fit we obtain a confidence interval of $\pm 0.5$ for $D$.

From Figure 11 and Table 2 we ascertain that as we had previously predicted, the clusters still obey the scaling pattern for all these different values for $p$. In fact, their fractal dimensions are still similar to the original model and from Figure 11 we notice that the linear fits are very similar but perhaps for low cluster sizes skewing their values. The radii changes here have been minimal and we note that DLA can maintain its stability when it incurs minor perturbations in its modelling. We investigate as to whether making more drastic changes to the particle radii disrupts the scaling pattern of DLA.

Instead of assigning a probability $p$ we now apply some distribution for the radii of particles. We consider radii normally, exponentially, gamma and uniformly distributed. In this model radii are not limited to taking only two values and in fact may take quite large radii. As remarked in our models section we set that the smallest radius of a particle to one in order

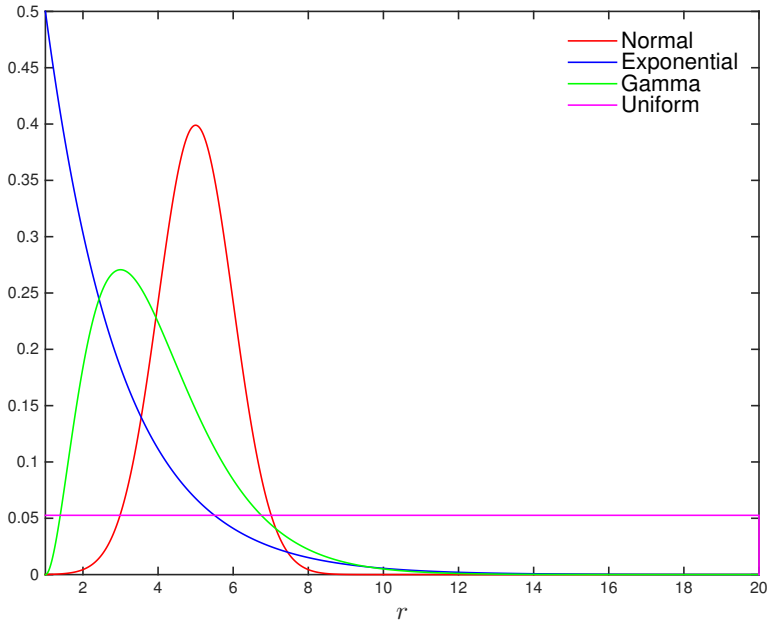



Figure 12: Distributions used for particle radii: (a) $\mathrm{N}(5,1)$, (b) $\mathrm{Exp}(\frac{1}{2})$, (c) $\Gamma(3,1)$ and (d) $\mathrm{U}[1,20]$.

to keep the model functioning. This is naturally done in an exponential, gamma or uniform distribution whose supports can be manipulated so that they only take values greater than or equal to one. The normal distribution has the real numbers as its support and so we use parameters such that obtaining a radius less than one is extremely unlikely and in the case it does happen we set the radius as one instead.

The following distributions were used: Normal Distribution with mean five and standard deviation one, Exponential Distribution with mean two, Gamma Distribution with shape parameter three and rate parameter one and Uniform Distribution on $[1,20]$. Figure 12 displays graphically the probability distributions of the four distributions used. A particle may be assigned a very large radius and as discussed above the possibility of large radii increases our computation time since at every step we must assume that we are close enough to aggregate to a particle of large radii. We implement this by assigning a maximum radius, $r_{max}$, where the probability of assigning a radius greater than this is negligible. For the normal distribution we use $r_{max} = 15$ and for the exponential, gamma and normal distribution, $r_{max} = 20$. When a particle is aggregated as before we update our Distance Matrix with new entries but instead modify the Vicinity Matrix we use to represent the size of the particle aggregated. To implement this we round the particle radius to the nearest integer, $r$, and our Vicinity Matrix is a $2L_{max} + r + 1$ square matrix containing the rounded-down distances between a cell and the edge of a disc of radius $r$ centred at the middle of the matrix. Figure 13 gives an example Vicinity Matrix where $L_{max} = 6$ and $r = 3$. The slight change here is that now the Distance Matrix contains the rounded-down distances between the edge of the closest particle in the cluster rather than the centre of the particle. With this implementation in place we can build clusters up to $75,000$ particles with particle radii varying from 1 to 20 in under half an hour for the normal, exponential and gamma distribution. For the uniform

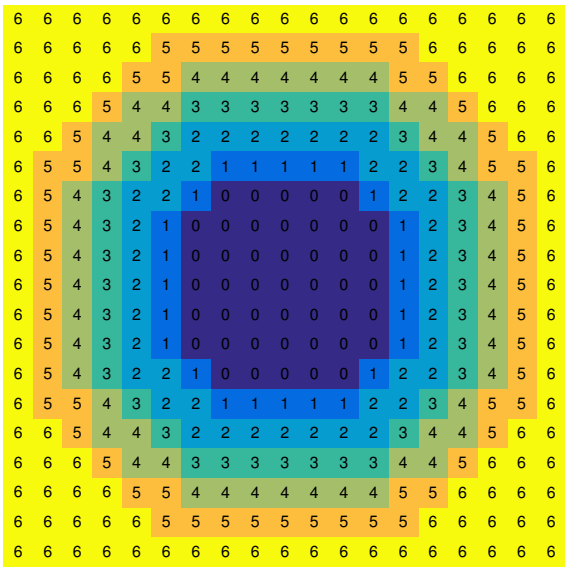


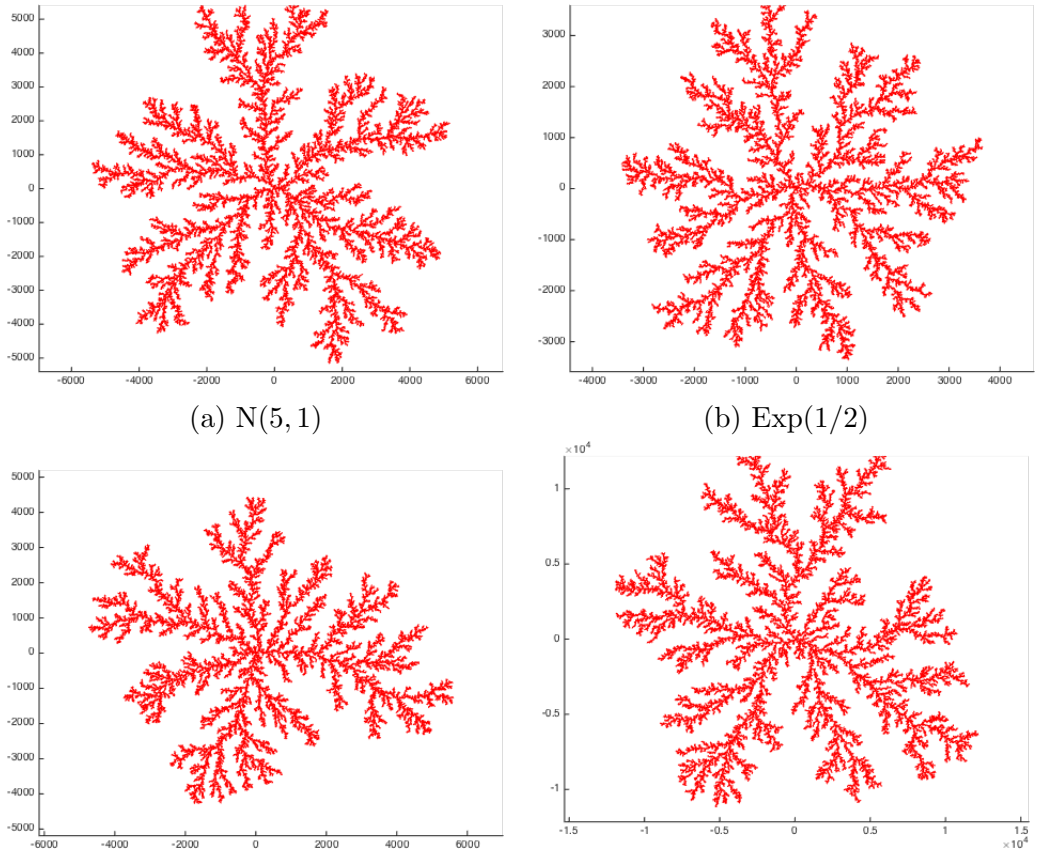

Figure 13: Modified Vicinity Matrix with $L_{max} = 6$ and $r = 3$.

| Distribution | N(5, 1) | Exp(1/2) | Γ(3, 1) | U[1, 20] |
|---|---|---|---|---|
| $D$ | 1.73 | 1.77 | 1.66 | 1.75 |

Table 3: Fractal Dimensions for clusters that takes particle radii according to different distributions.

distribution which sees large particles often it takes just under two and a half hours.

Clusters ranging from size 100 to 75, 000 were built and a log-log graph produced as before to examine whether this version DLA scaled as before. Figure 14 exhibits clusters of 75, 000 particles for the different distributions and Figure 15 the log-log graph. From the clusters we can see that the changing radii size has had little effect on the appearance of the structure with long branches still producing deep fjords. For the normal, gamma and exponential distribution this is not too surprising as the variance in each of these distributions is relatively small and hence the particle radii are mainly close to the mean value in each distribution. However the uniform distribution has a large variance and produces particles ranging from very small to very large often. Noticeably it has produced a significantly larger cluster than the others as expected but has not disrupted the appearance or stability of DLA. From our log-log graph we see that our cluster still appears to very much display a power scaling law since a straight-line fits the data well. The linear fits are all near parallel giving similar fractal dimensions as seen in Table 3. It was also observed from clusters that new arms are often built as a result of a particle of large radius. This is particularly evident in the exponential and gamma case where particles of large radius are unlikely and so occur rarely. Therefore when one does occur it is mostly followed by smaller particles which can attach to it from anywhere on its circumference and hence form several chains off it. This can be seen in Figure 16 where we have zoomed in on a section of the 75, 000 clusters built with exponentially and uniformly distributed particles.



(a) N(5, 1) (b) Exp(1/2)

(c) Γ(3, 1) (d) U[1, 20]

Figure 14: Clusters of 75, 000 particles for various particle radii distributions.

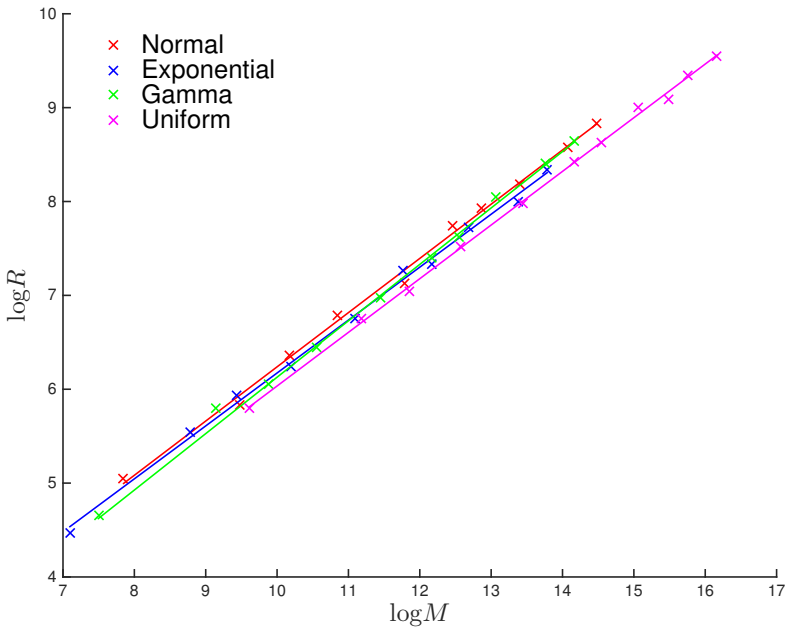
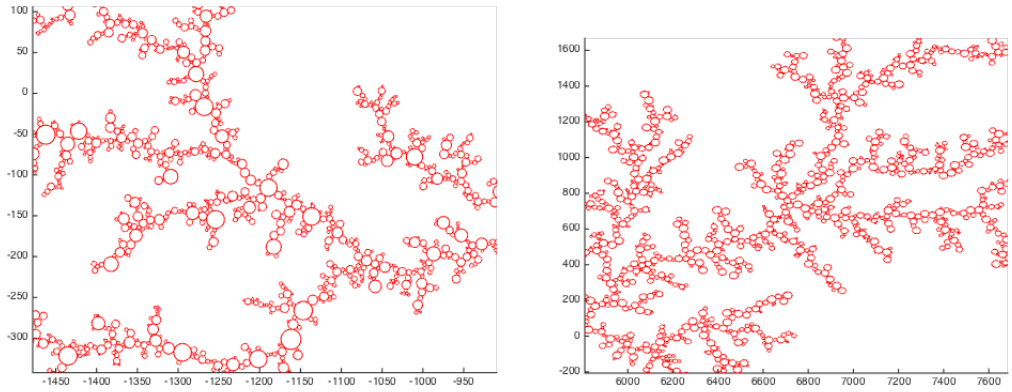



Figure 15: Growth scaling for clusters where particle radii follow distributions (a) N(5, 1), (b) Exp(1/2), (c) Γ(3, 1) and (d) U[1, 20].



(a) Exponential Distribution (b) Uniform Distribution

Figure 16: Close up segment of cluster built with exponentially and uniformly distributed radii displaying branch growth off larger particles.

# 4 Particle Attraction and Flow Affecting Stability

On a large scale, varying particle radii has had a minimal effect disturbing the fractal dimensions slightly but preserving the structure generally. We have built clusters up to $75,000$ particles which consist of varying particle sizes and have previously assumed that despite their varying size all these particles act uniformly in that they have no effect on the Brownian motion of the walker.We now begin to apply attraction between particles and combine this with varying particle sizes where large particles give greater attraction.

Attraction could be defined in numerous ways and so we need declare how we represent it. We want to introduce biased Brownian motion that has a preferred direction and we do this by introduction of bias in the exit distribution. Continuing with our model above we had previously been generating a uniform random angle on $[0, 2\pi)$ to decide our walking step direction. We modify the distribution of the angle to produce a bias in certain directions in order to produce this attraction. We can utilise the Beta distribution, which is a distribution on $[0, 1]$ with two shape parameters. If the two parameters are set to be equal then we produce a symmetric distribution around 0.5 which resembles a normal distribution shrunk to the unit interval. The special case where the shape parameters are both one in fact produces the Uniform Distribution on the unit interval. Figure 17 displays the distribution with various equal valued shape parameters. We are forced here to make some choice on how to represent attraction without precise knowledge of how it occurs in nature. The Beta distribution has a natural symmetric appearance on a fixed interval and is chosen both by natural intuition and the ease with which we can manipulate it. To utilise it in our model we alter the support of the function so that if we have a preferred walk in a direction of angle $\theta$ then the distribution takes values in $[\theta - \pi, \theta + \pi)$ with the distribution symmetric above $\theta$ as it was about 0.5 before. We then require an attraction parameter, $P$, to represent how much force the walker feels in the direction of $\theta$.

Before we begin the above investigation we start with a more general form of attraction to let us see the different effects of particle attraction. The effects of drift on DLA have

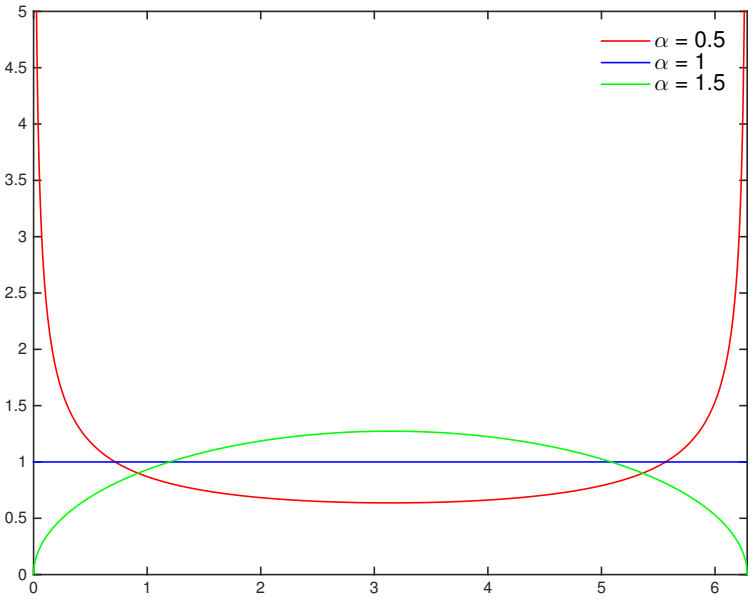
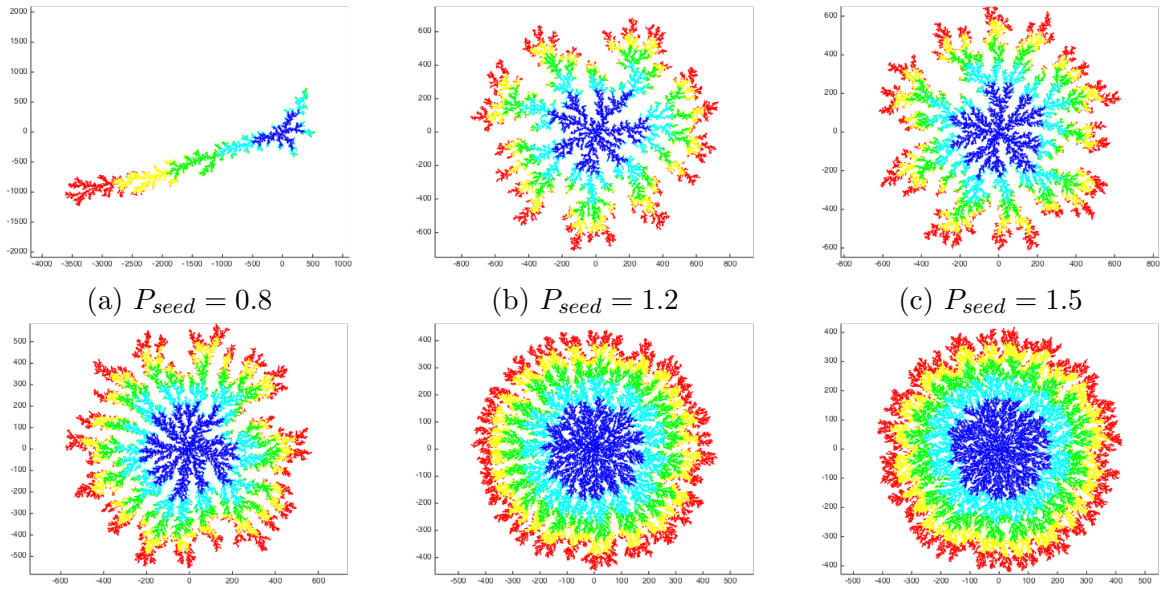



Figure 17: Beta$(\alpha, \alpha)$ distribution stretched over the interval $[0, 2\pi)$ for values $\alpha = 0.5$, 1 and 1.5.

been previously investigated [13] where it was shown that a model would generate a growth tendency in the opposite direction of the drift. We begin simply by applying an attraction to the seed for any walker. As described above the walker will feel an attraction to walk in the direction of the centre of the cluster governed by a Beta Distribution. We set the pulling parameter, $P_{seed}$, to be the equal parameters of the Beta Distribution. For $P_{seed} = 1$ we return to our original model with no pull in any direction, for $P_{seed} > 1$ the walker will experience a pull towards the seed and for $P_{seed} < 1$ a repulsion from the seed. When $P_{seed} \geq 1$ then we can build clusters as quickly as our original model and for $P_{seed}$ large enough in fact quicker since our particles aggregate quicker due to their pull towards the cluster. However for $P_{seed} < 1$ since the overarching force is in the opposite direction of the cluster it takes longer to build, with a cluster of size $50,000$ and $P_{seed} = 0.8$ taking a little under an hour and a half.

With a drift applied in any direction we quickly lose our DLA structure and produce clusters that look quite dissimilar to DLA. This can be seen in Figure 18 which exhibits clusters of size $50,000$ grown for numerous values for $P_{seed}$. For $P_{seed} = 0.8$ we no longer have resemblance to DLA with particles seeking to travel in the opposite direction of the seed and as a result the cluster produces one long arm to maximise distance from the seed. For large values we drift away from the recognisable DLA structure to a structure that is much more dense. This new structure bears a striking resemblance to clusters built by Ballistic Aggregation (BA), a model closely related to DLA. In this model the Brownian motion walk of the particles in DLA is replaced by ballistic trajectories. The particles are given initial velocities and if two collide they join and form a larger particle moving on with a velocity governed by the conservation of momentum. It is known that BA produces non fractal clusters with nontrivial scaling properties [10] and forms a model for the coalescence of aerosol particles in space. With our central force pulling particles to the seed we exhibit a crossover from



(a) $P_{seed} = 0.8$ (b) $P_{seed} = 1.2$ (c) $P_{seed} = 1.5$

(d) $P_{seed} = 2$ (e) $P_{seed} = 5$ (f) $P_{seed} = 10$

Figure 18: Clusters of $50,000$ particles where particles where subject to a central force of magnitude $P$ where colours indicate arrival times.

DLA to BA and lose our scaling properties and fractal tendencies. As we increase the attraction even further our cluster begins to resemblance clusters built by the Eden Growth model where clusters grow by random accumulation of material on their boundary and this provides a model for the growth of bacterial colonies. Further investigations can be made in this direction varying force direction and with the strength of the force depending on distance. Analysis has previously been done by Kim *et. al.* [8, 7] on lattice.

Applying a universal force to the seed has managed to distort the recognisable structure of DLA and now instead of applying a flow in a defined direction we utilise attraction between particles. We model this by considering the particle attraction negligible if the particle is outside a certain distance, the attraction distance, $D_a$. When a walker is within this distance to any aggregated particle then all aggregated particles near enough will exert some attraction on the walker. At this point a walker could be subject to multiple attractions from various aggregated particles and we must combine the forces felt on the walker to produce the direction of travel and calculate how much attraction is felt in this direction. This is done by calculating the weighted centre of mass in space of these particles by applying weights to particles. For each sufficiently close aggregated particle we attach a weight of one over its distance from the walker squared multiplied by the particle mass. We define this weight by analogy with gravitational and electrostatic forces in nature. With this, closer aggregated particles have a greater attraction and larger particles also have a greater attraction and we have weights $w_i$ such that

$$w_i = \frac{r_i^2}{\delta_i^2},$$

where $\delta_i$ is the distance between the walker and the $i^{th}$ aggregated sufficiently close particle

and $r_i$ is the radius of said particle. The centre of mass can then be calculated as

$$\mathbf{R} = \frac{1}{W}\left(\sum_i w_i x_i \ , \ \sum_i w_i y_i\right),$$

where $x_i$ and $y_i$ relate to the co-ordinates of each aggregated particle, the summations run over all sufficiently close particles and $W = \sum_i w_i$ is the total weight. Once we know the centre of mass we can determine the angle between the walker and the centre of mass. The walker then steps with a preference in this direction determined by a Beta distribution as above. The strength of this attraction, or equivalently, the equal parameters used in the Beta distribution, is decided by distance between the walker and the centre of mass and the total weight acting through the centre of mass, $W$. We derive an appropriate value for $P$ by stating some simple properties that we require of it. $P$ need change with the distance from the centre of mass squared and with the weight of the centre of mass linearly. We also require $P$ near one for a small weight and large distance. We let the equal parameter used in the Beta distribution, $P$, be equal to

$$P = 1 + \frac{W}{W_c}\left(1 - \left(\frac{\delta_R}{D_a}\right)^2\right),$$

where $\delta_R$ is the distance between the centre of mass and the walker, $D_a$ is the attraction distance and $W$ is the total weight of all sufficiently close particles as above. $W_c$ is a constant that we need define to obtain a suitable ratio for the weight of the centre of mass. It follows that $W_c$ should depend on the expected amount of weight of the centre of mass. Therefore we let

$$W_c = \frac{1}{K}{\mu_r}^2\left(D_a^2 - 1\right),$$

where $\mu_r$ is the expected radius of a particle in the cluster for some constant $K$. It can be seen that both $P$ and $W_c$ satisfy the respective properties we stated above. A small centre of mass far from the walker will tend to a Beta distribution of parameter one and hence result in no preferred walking direction. A large centre of mass close to the walker will tend to a Beta distribution of parameter greater than one, giving strong attraction to the centre of mass. We note that this value could be defined any number of ways and it may be of interest to look at whether alternative definitions will lead to different results. As we need use some definition, we continue with the ones derived above, confident that for sensible constants they will give us a suitable range of attraction.

We launch this calculation at every step where the walker is sufficiently close to the cluster, within the attraction distance. It was investigated as to whether instead of calculating at each step we assigned a matrix where each cell its preferred walking direction and attraction level and then at step approximated the preferred direction by taking information from the nearest integer cell. With this method whenever a particle is aggregated each sufficiently close cell would be recalculated to produce a new walking direction and attraction level. However this method took significantly longer with a $1,000$ particle cluster taking four minutes and sixteen seconds compared to calculating at each step taking only thirty-one seconds for $D_a = 20$ and $K = 100$.

| $D_a$ \ $K$ | $10^2$ | $10^4$ | $10^6$ | $10^8$ |
|---|---|---|---|---|
| 20 | 1.71 | 1.65 | 1.58 | 1.77 |
| 50 | 1.75 | 1.69 | 1.79 | 1.96 |
| 100 | 1.72 | 1.71 | 1.99 | 2.00 |

Table 4: Fractal Dimensions of clusters built with attraction between particles for numerous values for $D_a$ and $K$.

| $D_a$ \ $K$ | $10^2$ | $10^4$ | $10^6$ | $10^8$ |
|---|---|---|---|---|
| 20 | 25 | 13 | 19 | 11 |
| 50 | 117 | 38 | 27 | 30 |
| 100 | 409 | 224 | 103 | 87 |

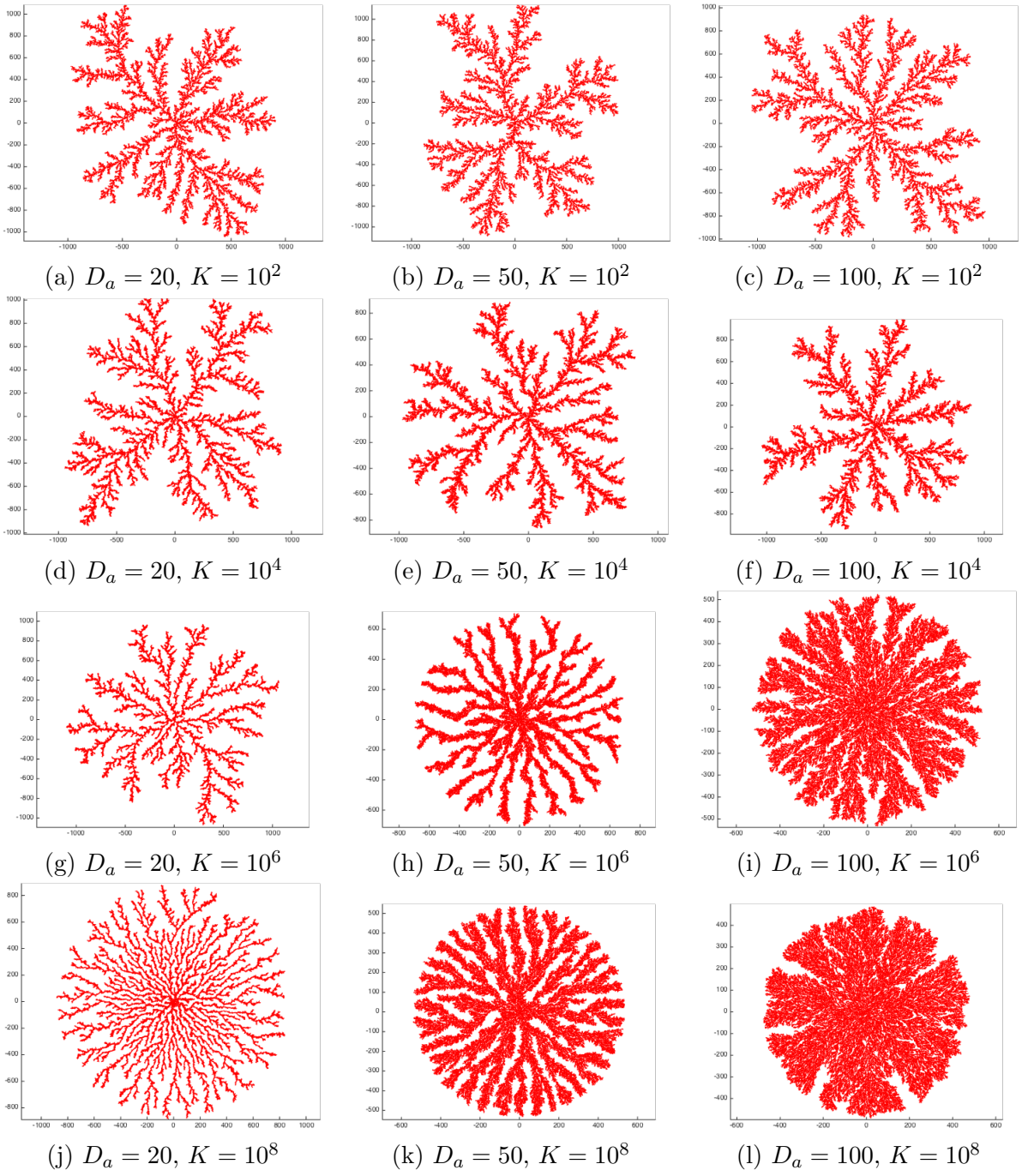
Table 5: Simulation times to the nearest minute for clusters of $75,000$ uniformly sized particles with attraction for numerous values for $D_a$ and $K$.

We begin by implementing this in the original model where particle radii are all uniform. In this the walker is simply attracted to each particle depending on their distance from the walker and as such the parameters $w_i$ and $W_c$ are simplified to

$$w_i = \frac{1}{\delta_i^2},$$
$$W_c = \frac{1}{K}\left(D_a^2 - 1\right),$$

since each particle has radius one. As constructed above, a particle closer than $D_a$ will apply some sort of attraction however large values of $D_a$ will increase computation time. By increasing $D_a$ we increase the number of times we have to launch the calculation to compute the total attraction and at each computation we need check a larger disc around the walker. Various values for $D_a$ and $K$ were tested with the fractal dimensions they produce noted in Table 4 and the computation times of a $75,000$ particle cluster noted in Table 5. An increase in $D_a$ leads to a drastic increase in computation time but greater $K$ leads to a decrease in computation time. This is due to large $K$ increasing the weight ratio in $P$ and hence increasing the attractive force so walkers aggregate faster. The clusters of size $75,000$ particles are exhibited in Figure 19.

The log-log graphs of radius against mass for the various $K$ values can be seen in Figure 20. We note from Table 4 that increasing $D_a$ generally results in an increase in fractal dimension and for high values of $D_a$ and $K$ we obtain very high fractal dimensions close to two. By observing our $75,000$ clusters we can see that the centre, or nuclei, of fractals with large values for $K$ are extremely dense. In early cluster formation branches have not yet formed and a production of a rough dense circle occurs giving us our high fractal dimension. The clusters with $D_a = 100$ and $K = 10^6$ or $K = 10^8$ can be seen to only just breaking away from their nuclei and thus for these parameters we build larger clusters, seen in Figure 21,



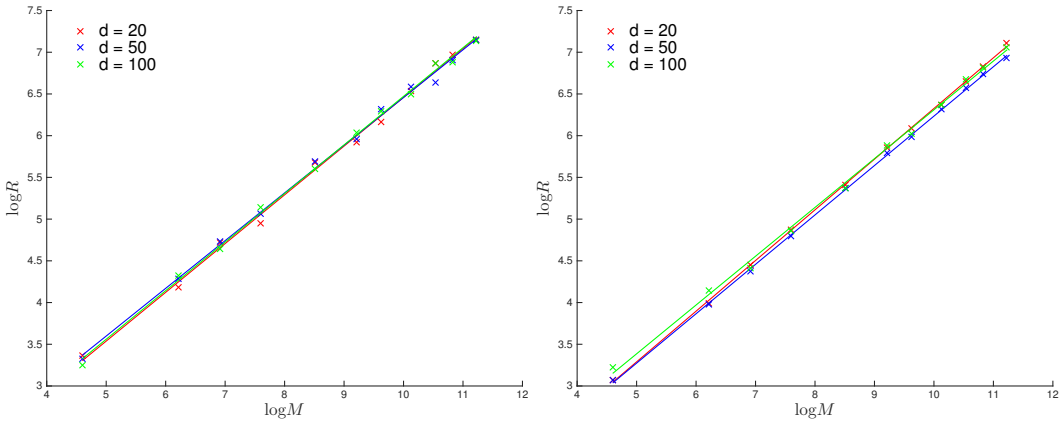
(a) $D_a = 20$, $K = 10^2$ (b) $D_a = 50$, $K = 10^2$ (c) $D_a = 100$, $K = 10^2$

(d) $D_a = 20$, $K = 10^4$ (e) $D_a = 50$, $K = 10^4$ (f) $D_a = 100$, $K = 10^4$

(g) $D_a = 20$, $K = 10^6$ (h) $D_a = 50$, $K = 10^6$ (i) $D_a = 100$, $K = 10^6$

(j) $D_a = 20$, $K = 10^8$ (k) $D_a = 50$, $K = 10^8$ (l) $D_a = 100$, $K = 10^8$

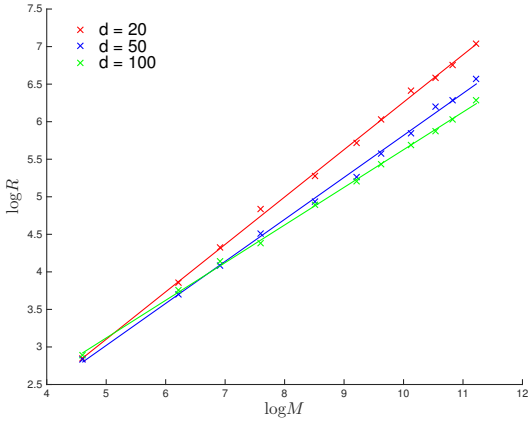
Figure 19: Clusters of $75,000$ where particles applied an attractive force on each other for numerous values for $D_a$ and $K$.



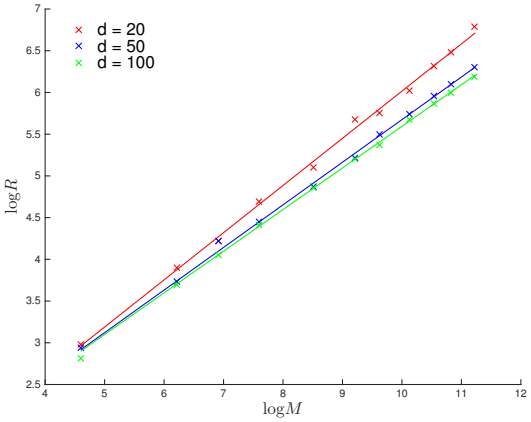

(a) $K = 10^2$




(b) $K = 10^4$

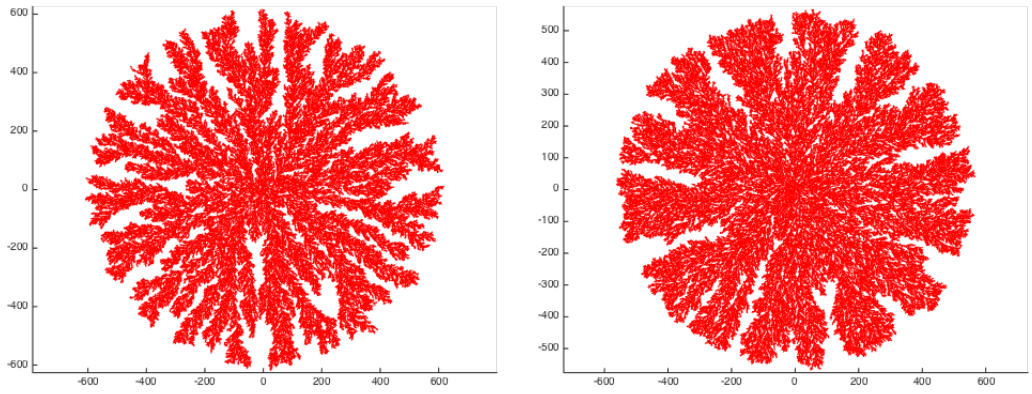



(c) $K = 10^6$




(d) $K = 10^8$

Figure 20: Log-log graph of radius against mass for the different values of $K$ and $D_a$.



(a) $K = 10^6$ (b) $K = 10^8$

Figure 21: Cluster of $10^5$ particles with attraction to the seed and parameters $D_a = 100$ and $K = 10^6$ or $K = 10^8$.

to see whether they will break away and form branches as hypothesised. As expected, if we increase the size of the cluster enough then the branching effect eventually appears and the cluster leaves its nuclei appearance.

Once branches have been produced we still have a severe lack of freedom in our system as large parameter values have produced highly symmetric and circular clusters of high density. The deep fjords are still produced but, while for low parameter values and in our original model these fjords did not have a uniform shape, now the fjords mostly have the appearance of thick lines from the outer edge to the nuclei. Similarly, the thickness of branches also increases for larger parameter values with low values exhibiting wiry thin branches and high parameters giving thick bushy ones. The width of these branches was measured to be roughly equal to the size of $D_a$ for large $K$ values. We note that these clusters hold a resemblance to ones built by Alves *et. al.* [14] who investigated the crossover between DLA and BA where walkers follow locally persistent trajectories.

The radius of gyration for a cluster is defined as

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

where we sum over all particles in the cluster and $r_i$ is the radius of the $i^{th}$ particle. Larger overall attraction has given us thicker more symmetric looking clusters and we explore the distribution about the seed of the fractals by calculating the radius of gyration in each direction and plotting this in Figure 22. This calculation is performed by taking small segments around the seed and calculating the radius of gyration from all particles whose centre of mass lies in the segment. For larger parameter values we near the appearance of a circle since we have relatively few arms and these arms are extremely thick. For low attraction distances but high attraction we obtain an extremely symmetric but spiked polar plot since we have a high number of arms which are all roughly equivalent in size. As our attraction decreases then our polar plots reveal non-symmetric, jagged and irregular

| $K$ | $10^4$ | $10^6$ | $10^8$ |
|---|---|---|---|
| $D$ | 1.72 | 1.61 | 1.80 |

Table 6: Fractal Dimensions, $D$, for a cluster that takes particle radii two with probability a half and radius one otherwise for various $K$ values.

plots since we have little symmetry in our clusters and the arms do not have a uniform appearance.

We now reintroduce particles of different radii beginning, as in the previous section, with clusters built of two distinct particle radii. We set $p = 0.5$, so that we release particles of radii one and two with equal probability. With different particle sizes now present in the cluster a walker will feel a greater attraction to larger particles and at a larger distance. As such we need $D_a$ to vary depending on the size of the particles. To combat this issue we define that a particle will apply an attraction a distance proportional to its radius away. This continues our above analogy since we consider any attraction force below a certain value negligible and hence the distance at which we fall below this value will be proportional to the square root of the mass. If the walker has radius one then an aggregated particle of radius one will apply attraction up to a distance of $D_{unit}$ and if the aggregated particle has radius two then this distance becomes $2D_{unit}$. Similarly if the walker has radius two and the aggregated particle has radius two then the attraction distance becomes $2^2 D_{unit}$. Generally if the walker has radius $r_w$ and the aggregated particle has radius $r_a$ then

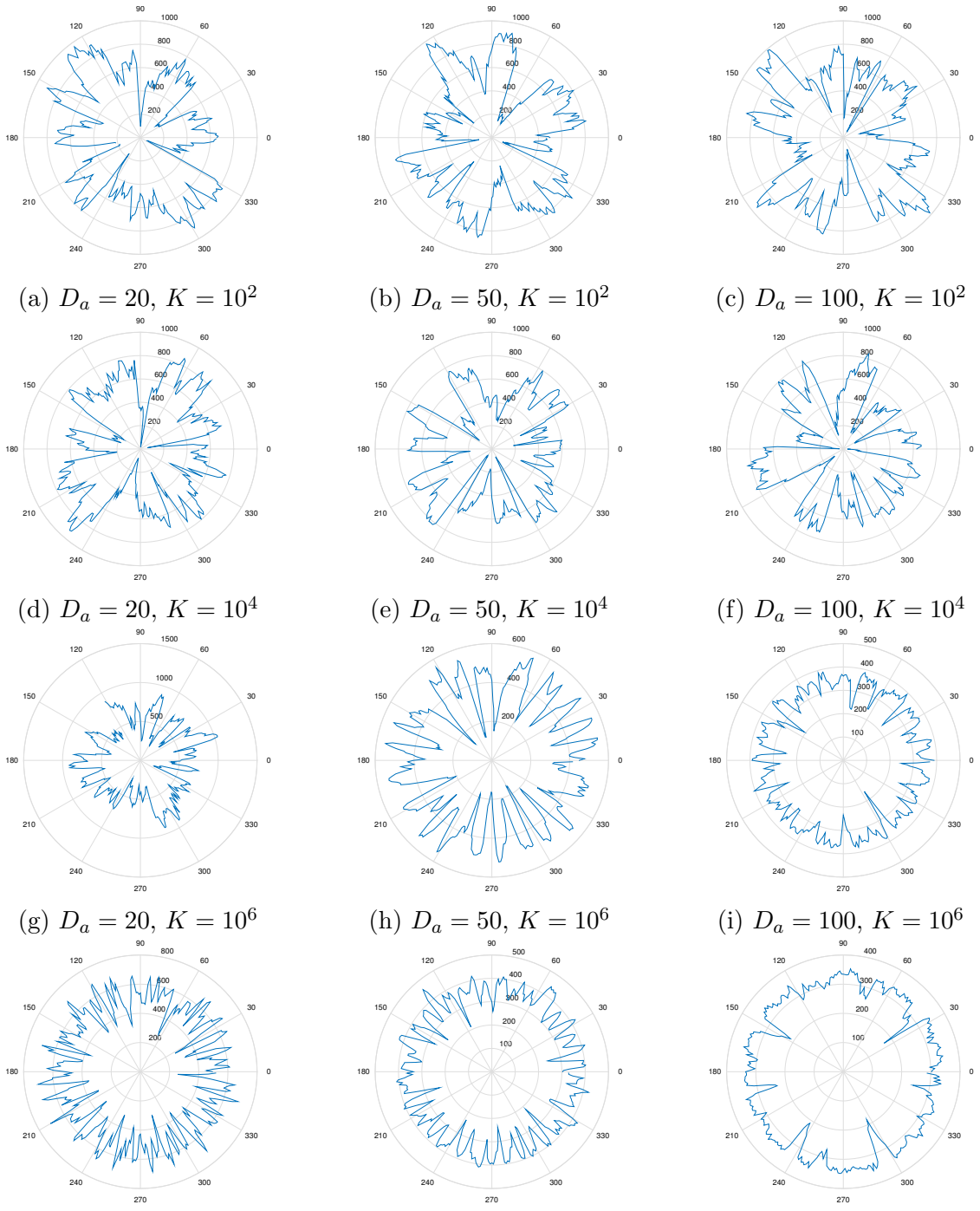
$$D_a = r_w r_a D_{unit}$$

where we set $D_{unit}$ as the attraction distance between two particles of unit radius. We then alter our definition of $W_c$ slightly to

$$W_c = \frac{1}{K}\mu_r^2 \left(D_{max}^2 - 1\right)$$

where $D_{max}$ is the maximum attainable attraction distance. As before we have two constants to vary, $D_{unit}$ and $K$. We set $D_{unit} = 20$ which gives a maximum attraction radius of eighty for two particles of radius two. Clusters were built with varying values for $K$ with the aggregates of $75,000$ particles in Figure 23, our log-log graph in Figure 24 and the fractal dimensions in Table 6. Previously a linear best approximation had fit the data points well but now we note that, even though still a good fit for the data, our points do not lie as well on the linear approximation, especially for the lower attraction values. Our linear fits are more spread than with uniformly sized particles and lower attraction has produced consistently larger clusters. As before higher attraction gives a larger fractal dimension however moderate attraction has given a lower fractal dimension. From our $75,000$ clusters we observe that this is due to moderate attraction reducing the freedom in the system and forming less complex branches however with high attraction we have a severe lack of freedom but a large number of arms dominating the cluster.

Continuing with above we now release particles of radii governed by a continuous distribution. Particles range in size from one to twenty and, as with two radii sizes, we need vary $D_a$ with



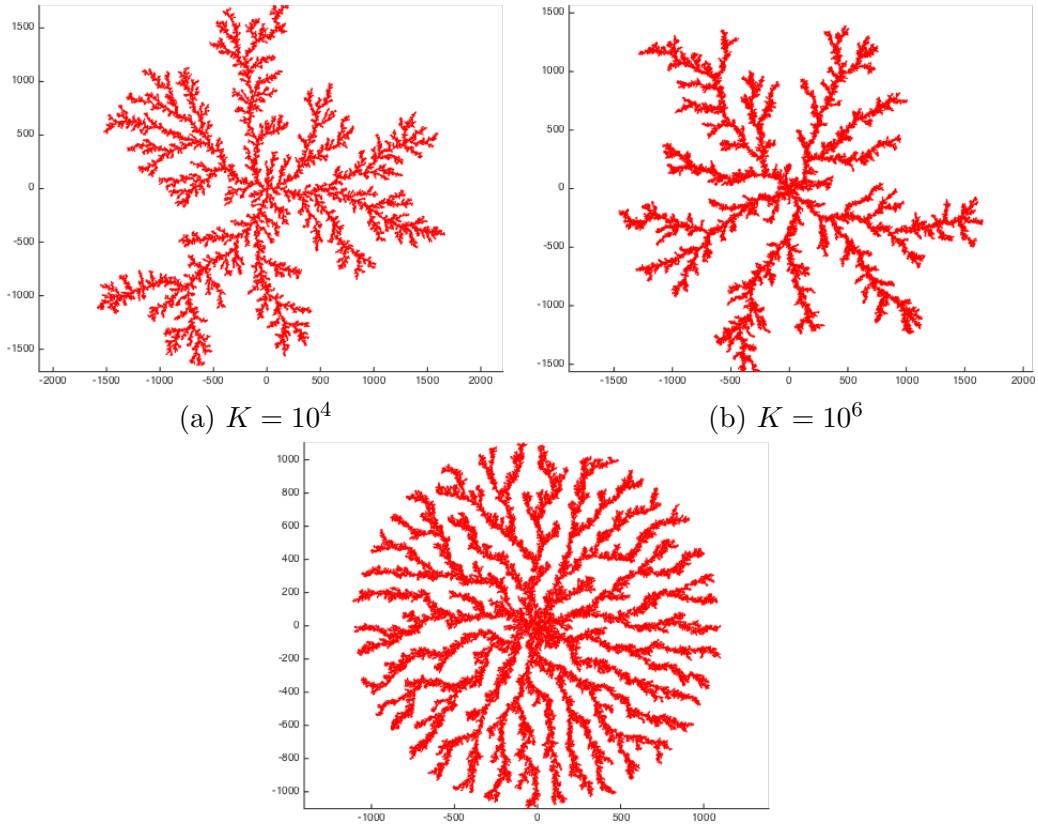
(a) $D_a = 20$, $K = 10^2$ (b) $D_a = 50$, $K = 10^2$ (c) $D_a = 100$, $K = 10^2$

(d) $D_a = 20$, $K = 10^4$ (e) $D_a = 50$, $K = 10^4$ (f) $D_a = 100$, $K = 10^4$

(g) $D_a = 20$, $K = 10^6$ (h) $D_a = 50$, $K = 10^6$ (i) $D_a = 100$, $K = 10^6$

(j) $D_a = 20$, $K = 10^8$ (k) $D_a = 50$, $K = 10^8$ (l) $D_a = 100$, $K = 10^8$

Figure 22: Radius of gyrations in each direction around the seed particle.



(a) $K = 10^4$ (b) $K = 10^6$

(c) $K = 10^8$

Figure 23: Cluster of $75,000$ particles built with particles taking radii one half the time and two otherwise exhibiting inter-particle attraction with $D_{unit} = 20$ and various values for $K$.

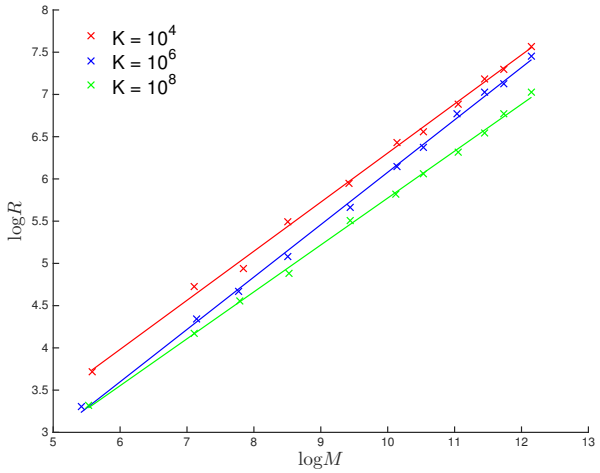



Figure 24: Log-log graph of radius against mass for clusters built with particles given radius two or one with equal probability for various values for $K$.

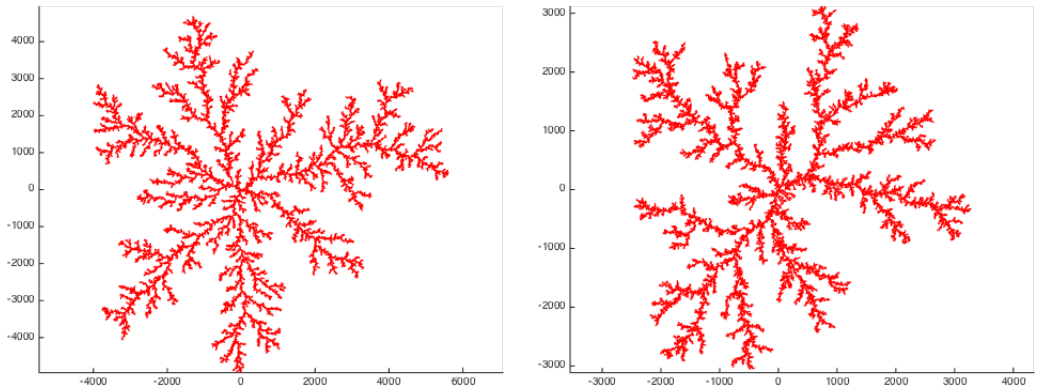
| Distribution | N(5, 1) | Exp(1/2) | Γ(3, 1) | U[1, 20] |
|---|---|---|---|---|
| $D$ | 1.67 | 1.67 | 1.76 | 1.79 |

Table 7: Fractal Dimensions for clusters that takes particle radii according to different distributions and have inter-particle attraction.

particle radii. Using the same model as with two radii then a walker of size twenty could be attracted to an aggregated particle of size twenty at a distance of $20^2 D_{unit}$ now. We use the same definition of $W_c$ used above with two radii. As above we need define our variable $K$. We seek a sensible value for $K$ that gives intermediate attraction levels because, as we have seen before, high $K$ values gives overpowering attraction while low $K$ will give virtually none. For $K = 10^9$ we obtain attraction levels ranging from one to no higher than a hundred giving us a sensible spread of attraction. Large particles give an extremely large attraction distance and this proves costly on our computation time since now at most steps we feel attraction and have to calculate the overall preferred direction of travel and attraction level about a much larger circle round the walker. Indeed at each step we must assume that a particle of maximum radius is acting on it from as far away as possible. This renders computation times very large and therefore we implement an additional method to quicken the simulation.

We create a new matrix, the Attraction Matrix, which will be an on-lattice version of the cluster where each cell contains the radius needed by the walker to feel attraction. To calculate this value each time a particle is aggregated we update a square matrix around the aggregated particle of size $2D_{unit} r_a r_{max} + 1$ where $r_a$ is the radius of the aggregated particle and $r_{max}$ is the maximum particle radius. We know that a walker will feel attraction from the aggregated particle up to a distance $D_{unit} r_a r_w$ where $r_w$ is the radius of the walker. We can calculate the distance of each cell from the middle and thus enter in the cell the radius needed by the walker to feel attraction at this point. Then at each point the walker can look into the cell relating to its nearest integer coordinates and if the walker has a radius greater than the radius needed we only then calculate the attraction. Ultimately, while this saves us considerable time in the long-run, this simulation is still a time consuming construction. As the size of particles have increased, we increase the minimum step size to $L_{min} = 0.5$ in order to help improve efficiency. We can then build clusters up to size $50,000$ in just under one and a half hours in the normal, exponential and gamma case. However for the uniform distribution seeing large particles often leads to a drastic increase in computation time and building a cluster of $50,000$ particles takes just over twenty nine hours. The clusters of size $50,000$ are given in Figure 25, their log-log graphs in Figure 26 and the fractal dimensions produced in Table 7.

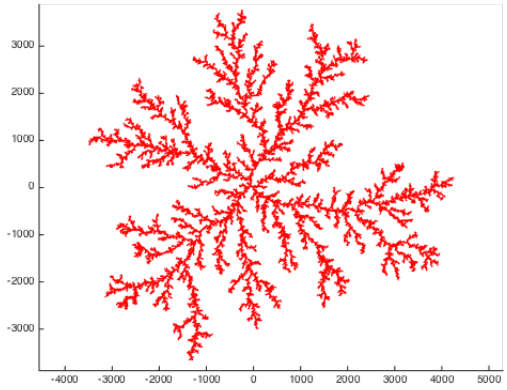
With such small variance in particle size the normal distribution gives a cluster similar in appearance to our original model since most particles have nearly equal size. As the variance of the distribution increases the complexity of the $50,000$ cluster disappears with the exponential case displaying first signs of a loss of freedom. With its high variance the uniform distribution has produced a cluster reminiscent of higher attraction values with symmetry and lack of freedom both evident in the cluster and equally gives a larger fractal



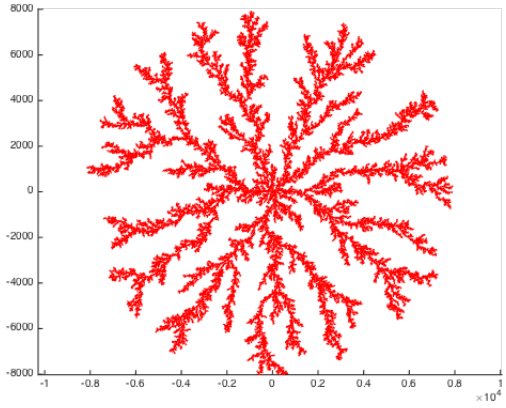
(a) N(5, 1)



(b) Exp(1/2)



(c) Γ(3, 1)



(d) U[1, 20]

Figure 25: Cluster of 50, 000 particles built with particles taking radii according to distributions exhibiting inter-particle attraction with $D_{unit} = 2$ and $K = 10^9$.

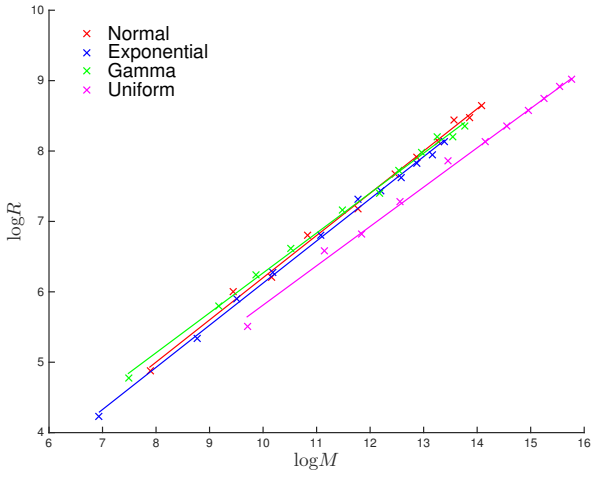



Figure 26: Log-log graph of radius against mass for clusters built with particles given radius governed by different continuous distributions with inter-particle attraction.

dimension than the others. Increasing $K$ further will increase our attraction so that at every step we force the particle in a certain direction and ignore much of the effect of particle size. For these high values we produce clusters similar in appearance to those with two radii and uniformly sized particles with high $K$.

# 5 Conclusion

We have investigated the scaling and structural properties of DLA when particles have been allowed to take varying particle sizes depending on both continuous and discrete distributions. We have then applied inter-particle attraction by introducing a bias in the particles direction of travel governed by a beta distribution with equal parameters.

Altering the radii of particles had little effect on the overall structure of DLA and we continued to see similar clusters as to when particles had uniform size in both the discrete and continuous case. With two distinct radii the fractal dimension incurred a slight increase when we had equal probability of either size. When particle radii were distributed we equally noted an increase in fractal dimension, although there appeared to be no relation between the variance of the distribution and the fractal dimension. This stability is an important feature since DLA provides a model for various systems across numerous scientific fields. In these scenarios we are limited by how our understanding of the processes which occur in natural systems. Effective DLA models should therefore not rely on these exact details which are often lacking. We have shown here that this is possible for varying particle sizes in that small perturbations in the construction do not affect the overall formation and our results remain structurally consistent.

However, when particle attraction or flow was applied we quickly lost our recognisable features. With a flow applied to the seed particle, as the magnitude of this flow increased we experienced a crossover from DLA to BA, with perhaps a further crossover to the Eden Model for even greater flow. Particle attraction was defined before it was implemented by setting out some basic rules for it to follow and then observing that our definition obeyed these. Our rules were proposed by attempting to mirror naturally occurring attraction scenarios however it would be of considerable interest to see whether alternative definitions of attraction would produce similar clusters. High particle attraction over large distances produces similar results to those observed under central flow, with the formation of a structure consisting of a dense nuclei and then long thick arms of uniform appearance positioned symmetrically about the nuclei. By observing the radius of gyration in each direction we were able to observe numerically the transformation in appearance of the clusters and the changes towards increased symmetry and a more uniform appearance.

When applying particle attraction between particles of different radii we chose an appropriate value for $K$ so that the attraction difference between small and large particles was significant. When clusters were formed with this we produced a rigidity in the structures. Arms were longer and straighter than previously and often appeared in opposite directions across the seed. With slight attraction this decreased the density of our clusters however, when we continued to increase $K$ the clusters grew symmetric and with an increasing number of

arms. Based on the initial results of our simulations we hypothesise that if the fractal dimensions were thoroughly examined we would note a decrease in fractal dimensions as gentle attraction is applied but then as greater attraction is applied an increase in fractal dimensions.

Optimisation was key throughout the all simulations and we were able to build the majority of clusters with all amendments in sensible time. However when particle sizes varied drastically and particle attraction was implemented, our definition of attraction produced drastically increased computation times, since at each step we needed to calculate the preferred direction of travel and attraction level. An alternative definition of attraction could potentially lead to a faster build and the optimisation of this problem is certainly an area for potential future investigation. Indeed the main obstacle in this calculation is that the attraction a walker feels depends on the size of the walker and so any pre-allocation of flow in the space would need to vary for each possible particle size. An alternative definition of attraction may ignore this dependence on the radius of the walker and thus an approximation of this may be possible by building a flow map so at each point in space the walker could instantly know the preferred method of travel. With this method you need update the map every time a particle is aggregated and this pre-allocation of flow would increase the speed of the system dramatically although perhaps lose some of the relevance to naturally occurring systems we have tried to maintain throughout the process.

Although no real life model motivated this research it has been observed that, similar to with particle attraction, the deposition of particles on fibres and surfaces has undertaken a crossover from a fractal to a uniform shape when the length scale has increased [13]. However we note that this crossover has been seen theoretically in various scenarios including when particles moved by a mix of brownian motion and ballistic paths [1] and biased random walks [6]. The formation of DLA is an extremely complex yet probabilistic model of which no comprehensive understanding has yet been achieved. To obtain trustworthy results many thousands of clusters are required. Time limitations have prevented us from obtaining a full compliment of results however we have identified some important areas for potential future research. Whether higher levels of attraction to the seed will induce a crossover to the Eden Model, whether our definition of attraction is sufficiently accurate and if alternative definitions might provide different results are all possible routes of further investigation.

# 6 References


[1] S. Alves and S. Ferreira Jr. Aggregation in a mixture of brownian and ballistic wandering particles. *Physical Review E*, 73(5):051401, 2006.

[2] S. Alves and S. Ferreira Jr. Is it really possible to grow isotropic on-lattice diffusion-limited aggregates? *Journal of Physics A: Mathematical and General*, 39(12):2843, 2006.

[3] R. Ball and R. Brady. Large scale lattice effect in diffusion-limited aggregation. *Journal of Physics A: Mathematical and General*, 18(13):L809, 1985.

[4] V. A. Bogoyavlenskiy. How to grow isotropic on-lattice diffusion-limited aggregates. *Journal of Physics A: Mathematical and General*, 35(11):2533, 2002.

[5] K. Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2004.

[6] S. Ferreira Jr, S. Alves, A. F. Brito, and J. Moreira. Morphological transition between diffusion-limited and ballistic aggregation growth patterns. *Physical Review E*, 71(5):051402, 2005.

[7] Y. Kim and K. Choi. Anisotropies in aggregates with biased random walks on two-dimensional lattices. *Physical Review E*, 48(2):1586, 1993.

[8] Y. Kim, K. Choi, and H. Pak. Aggregates with biased random walks on a square lattice. *Physical Review A*, 45(8):5805, 1992.

[9] 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.

[10] S. Liang and L. P. Kadanoff. Scaling in a ballistic aggregation model. *Physical Review A*, 31(4):2628, 1985.

[11] M. Matsushita and H. Fujikawa. Diffusion-limited growth in bacterial colony formation. *Physica A: Statistical Mechanics and its Applications*, 168(1):498–506, 1990.

[12] M. Matsushita, M. Sano, Y. Hayakawa, H. Honjo, and Y. Sawada. Fractal structures of zinc metal leaves grown by electrodeposition. *Physical review letters*, 53(3):286, 1984.

[13] P. Meakin. Effects of particle drift on diffusion-limited aggregation. *Physical Review B*, 28(9):5221, 1983.

[14] I. R. Nogueira, S. G. Alves, and S. C. Ferreira. Scaling laws in the diffusion limited aggregation of persistent random walkers. *Physica A: Statistical Mechanics and its Applications*, 390(23):4087–4094, 2011.

[15] E. Sander, L. M. Sander, and R. M. Zi. Fractals and fractal correlations. *Computers in Physics*, 8(4):420, 1994.

[16] L. M. Sander. Diffusion-limited aggregation: a kinetic critical phenomenon? *Contemporary Physics*, 41(4):203–218, 2000.

[17] S. Tolman and P. Meakin. Off-lattice and hypercubic-lattice models for diffusion-limited aggregation in dimensionalities 2–8. *Physical Review A*, 40(1):428, 1989.

[18] T. Witten Jr and L. M. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical review letters*, 47(19):1400, 1981.

[19] C. A. Yates and R. E. Baker. Isotropic model for cluster growth on a regular lattice. *Physical Review E*, 88(2):023304, 2013.

[20] R. Zhang, M. Hummelgård, and H. Olin. Size and concentration controlled growth of porous gold nanofilm. *physica status solidi (a)*, 209(3):519–523, 2012.

# A MATLAB Code

Included here is the MATLAB code used for building clusters with radii that are exponentially distributed with inter-particle attraction. This code is chosen as it utilises all the techniques we have used throughout the project including: the methods used for optimising the original algorithm, the use of several vicinity matrices and the use of the Attraction Matrix.

```
function [radius,clusterMass] = ExpDistAttraction(numberOfParticles);

% This is off-lattice DLA simulation with particle radii exponentially
% distributed. The particles have attraction for one another with larger
% particles exhibiting greater attraction.

startTime = tic;

%% SET UP VARIABLES AND MATRICES

% We begin by setting up some fixed variables for the simulation.

minStepSize = 0.5; % The minimum step size a particle can take.
maxInspectedDistance = 80; % The maximum inspected distance.
maxParticleRadius = 20; % The max radius of a particle.
unitAttractionDistance = 2;
% The attraction distance between two unit particles.
K = 10^9; % Attraction constant K.
expectedParticleRadius = 3; % Expected particle radius for exponential.
maxAttractionDistance = unitAttractionDistance*maxParticleRadius^2;
% The maximum attraction distance possible.
expectWeight = (expectedParticleRadius^2*(maxAttractionDistance^2 - 1))/K;
% Expected weight constant.


% To avoid having to create huge matrices since the cluster won't get as
% big as the number of particles we can use a rough estimate.
expectedRadius = expectedParticleRadius*numberOfParticles^(1/1.55);
maxRlaunch = expectedRadius + 5*minStepSize;
maxRkill = 5*maxRlaunch;
matrixSize = round(2.5*maxRkill)

% Here are variables that will change as the simulation progresses, but we
% set them up with initial values.

particleNumber = 1;
maximumDistance = 0;

% StuckParticles will store the exact positions of the particles in the
% cluster. The third column stores the radius of the stuck particle.
stuckParticles = zeros(numberOfParticles,3);

% The clusterMatrix will store an on-lattice version of our cluster.
```

```
clusterMatrix = zeros(matrixSize);

% The distanceMatrix contains the rounded down distances between the cell
% and the nearest occupied cell. Occupied cells are labeled as zero.
distanceMatrix = maxInspectedDistance*ones(matrixSize);

% The attractionMatrix contains the radii needed for the walker to feel
% attraction.
attractionMatrix = (maxParticleRadius+1)*ones(matrixSize);

%% SET UP VICINITY MATRICES UP TO THE MAXIMUM PARTICLE SIZE
% The vicinityMatrices contains the rounded down distances between the
% cells and the cell in the middle of the grid. We define them at the
% beginning and then never recalculate.

for i = 1:maxParticleRadius
    submatrixSize = 2*maxInspectedDistance + 2*i + 1;
    tempVicinityMatrix = zeros(submatrixSize);

    % Let us set up the vicinityMatrix for this radii size

    vicinityMiddle = maxInspectedDistance + i + 1;

    for k=1:submatrixSize
        for j=1:submatrixSize
            xdistance = abs(vicinityMiddle - j);
            ydistance = abs(vicinityMiddle - k);
            distanceFromMiddle = max(sqrt(xdistance^2 + ydistance^2)...
                - i,0);
            tempVicinityMatrix(j,k) = min(floor(distanceFromMiddle),...
                maxInspectedDistance);
        end
    end
    vicinityMatrices{i} = tempVicinityMatrix;

end


%% PLANT CENTRE SEED

% Let's start the simulation by planting a seed at the centre.
middle = round((matrixSize +1)/2);
stuckParticles(particleNumber,1) = 0;
stuckParticles(particleNumber,2) = 0; % seed is at the origin
stuckParticles(particleNumber,3) = exprnd(2) +1 ; % define seed radius
clusterMatrix(middle,middle)=particleNumber;

% Update the distanceMatrix using vicinityMatrix using method described in
% paper.
tempVicinityMatrix = vicinityMatrices{1};
submatrix = distanceMatrix(middle-maxInspectedDistance:middle+...
    maxInspectedDistance,middle-maxInspectedDistance:middle+...
    maxInspectedDistance);
for i = 1:2*maxInspectedDistance + 1
```

```
    for j=1:2*maxInspectedDistance + 1
        submatrix(j,i) = min(tempVicinityMatrix(j,i),submatrix(j,i));
    end
end
distanceMatrix(middle-maxInspectedDistance:middle+...
    maxInspectedDistance,middle-maxInspectedDistance:middle+...
    maxInspectedDistance) = submatrix;

% Update the attractionMatrix using method described in paper.

tempAttractionDistance = ...
    round(unitAttractionDistance*maxParticleRadius);
radiiNeededSubmatrixSize = 2*tempAttractionDistance + 1;
% Take a submatrix of the close enough values to update.
radiiNeededSubmatrix = attractionMatrix(middle - ...
    tempAttractionDistance:middle + tempAttractionDistance,...
    middle - tempAttractionDistance:middle + tempAttractionDistance);
% Define middle of this submatrix.
radiiNeededMiddle = ...
    tempAttractionDistance;
% For each cell calculate the radii needed for a walker to feel attraction
% from the seed. Update the entry of the cell to the minimum of the old
% cell value and this newly calculated radius.
for k = 1:radiiNeededSubmatrixSize
    for l = 1:radiiNeededSubmatrixSize
        xdif = abs(k - radiiNeededMiddle);
        ydif = abs(l - radiiNeededMiddle);
        distanceFromMiddle = sqrt(xdif^2 + ydif^2);
        % Calculate radii needed.
        radiiNeeded = ...
            distanceFromMiddle/unitAttractionDistance;
        % Update if this radii is smaller than previously.
        radiiNeededSubmatrix(l,k) = ...
            min(radiiNeededSubmatrix(l,k),radiiNeeded);
    end
end
% Put the updated submatrix back in.
attractionMatrix(middle - tempAttractionDistance:middle +...
    tempAttractionDistance,middle - tempAttractionDistance:middle +...
    tempAttractionDistance) = radiiNeededSubmatrix;

%% BEGIN LAUNCHING PARTICLES

% Now we have planted our seed at the centre of the matrix and our distance
% matrix is initially set up. We can now launch our particles from our
% launching radius.

endscript = 0; % Bool to determine when to end the script.

while endscript == 0

    % First we must launch the particle from the launching radius and
    % determine what radii it will have. We need to make sure the launching
    % radius is large enough so that we can take at least one step.
```

```
152     Rlaunch = maximumDistance + 2*maxParticleRadius + minStepSize;
153     randStartAngle = 2*pi*rand;
154     x = Rlaunch*sin(randStartAngle);
155     y = Rlaunch*cos(randStartAngle);
156
157     % Now determine the particle radius.
158     rparticle = exprnd(2) + 1;
159     if rparticle > maxParticleRadius
160         rparticle = maxParticleRadius;
161     end
162
163     % For this radius we define the hittingDistance, which is the distance
164     % between the centre of the particle and the nearest particle. The
165     % centreDistance is the smallest distance between the centre of the
166     % walker and the centre of a particle and the attractionDistance is the
167     % largest possible distance attraction will be felt.
168
169     hittingDistance = rparticle + minStepSize + 1;
170     centreDistance = rparticle + maxParticleRadius + minStepSize + 1;
171     attractionDistance = ...
172         maxParticleRadius*rparticle*unitAttractionDistance;
173
174     % Now we can begin our Brownian motion loop.
175     aggregate = 0; % Bool to break below Brownian motion loop.
176
177     while aggregate == 0
178
179         distanceFromCenter = sqrt(x^2 + y^2);
180         if distanceFromCenter > 5*Rlaunch
181             % If particle has gone outside the killing radius then we
182             % relaunch it from the launching radius.
183
184             randStartAngle = 2*pi*rand;
185             x = Rlaunch*sin(randStartAngle);
186             y = Rlaunch*cos(randStartAngle);
187
188         else
189             % The walker is not outside the killing radius so we need to
190             % find any attraction, check how far we can walk and whether we
191             % will collide.
192
193             % We extract the nearest distance from the distanceMatrix and
194             % the radii needed for attraction from the attractionMatrix.
195             nearestDistance = distanceMatrix(middle - round(y),...
196                 middle + round(x));
197             radiiNeededTemp = attractionMatrix(middle - round(y),...
198                 middle + round(x));
199
200             % If the walker has large enough radius we find the preferred
201             % walking direction and attraction level.
202             if (rparticle > radiiNeededTemp) || (rparticle ==...
203                     radiiNeededTemp)
204                 xround = middle + round(x);
205                 yround = middle - round(y);
```

```
206                 attractionCandidateMatrix = clusterMatrix(round(yround...
207                     - attractionDistance):round(yround...
208                     + attractionDistance),round(xround...
209                     - attractionDistance):round(xround...
210                     + attractionDistance));
211
212                 % This matrix contains the labels of stuck particles near
213                 % the walker that could apply attraction. We retrieve these
214                 % labels.
215                 attractionCandidates = [];
216                 for i = 1:2*attractionDistance+1
217                     for j = 1:2*attractionDistance+1
218                         % If there is an entry then add it to the
219                         % candidates.
220                         if attractionCandidateMatrix(j,i) ~= 0
221                             attractionCandidates = [attractionCandidates...
222                                 attractionCandidateMatrix(j,i)];
223                         end
224                     end
225                 end
226
227                 % Now we have the labels we can extract the exact
228                 % co-ordinates from stuckParticles. We have the current
229                 % position of the walker (x,y) as well as the positions of
230                 % all the candidates. We can then now find the preferred
231                 % angle of travel and attraction level.
232
233                 % Variables used to calculate the centre of mass and
234                 % weight.
235
236                 xcentermass = 0;
237                 ycentermass = 0;
238                 totalweight = 0;
239
240                 for t=1:length(attractionCandidates);
241                     % Retrieve the coordinates and radius of the particle.
242                     tempLabel = attractionCandidates(t);
243                     xtemp = stuckParticles(tempLabel,1);
244                     ytemp = stuckParticles(tempLabel,2);
245                     rtemp = stuckParticles(tempLabel,3);
246                     % Check if it is close enough to apply attraction.
247                     walkerdistance = sqrt((x-xtemp)^2 + (y-ytemp)^2);
248                     attractionTempDistance = ...
249                         rtemp*rparticle*unitAttractionDistance;
250                     % If it is then add it to the centre of mass
251                     % calculation.
252                     if walkerdistance < attractionTempDistance
253                         xcentermass = ...
254                             xcentermass + (rtemp/walkerdistance)^2*xtemp;
255                         ycentermass = ...
256                             ycentermass + (rtemp/walkerdistance)^2*ytemp;
257                         totalweight = ...
258                             totalweight + (rtemp/walkerdistance)^2;
259                     end
```

```
260                 end
261                 % In case no particle is close enough to apply attraction.
262                 if totalweight == 0
263                     attractionLevel = 1;
264                     angleToCentre = pi;
265                 else
266                     % Else calculate where the centre of mass is and so the
267                     % preferred walking direction.
268                     xcentermass = xcentermass/totalweight;
269                     ycentermass = ycentermass/totalweight;
270 
271                     xdif = x - xcentermass;
272                     ydif = y - ycentermass;
273                     xquad = sign(xdif);
274                     yquad = sign(ydif);
275                     quadcheck = xquad + yquad;
276                     if quadcheck == 2
277                         % in top right
278                         angleToCentre = 3*pi/2 - atan(ydif/xdif);
279                     elseif quadcheck == -2
280                         % in bottom left
281                         angleToCentre = atan(xdif/ydif);
282                     else
283                         if xquad == 1
284                             % in bottom right
285                             angleToCentre = 3*pi/2 + atan(abs(ydif)/xdif);
286                         else
287                             % in top left
288                             angleToCentre = pi/2 + atan(ydif/abs(xdif));
289                         end
290                     end
291                     distanceFromR = sqrt(xdif^2 + ydif^2);
292                     % Calculate attraction level.
293                     attractionLevel = 1 + ...
294                         (totalweight/expectWeight)*(1 - ...
295                         (distanceFromR/attractionDistance)^2);
296                     % In rare case that we produce attractionLevel below 1
297                     % we set it as 1 since we do not have repulsion in our
298                     % system.
299                     if attractionLevel < 1
300                         attractionLevel = 1;
301                     end
302                 end
303                 % Calculate angle walker will travel with attraction
304                 % acting.
305                 randAngle = ...
306                     betarnd(attractionLevel,attractionLevel)*2*pi ...
307                     + angleToCentre - pi;
308             else
309                 % If there was no attraction then the walker walks in any
310                 % direction uniformly.
311                 randAngle = 2*pi*rand;
312             end
313 
```

```
314             % Now check if the walker is close enough to aggregate.
315             if (nearestDistance < hittingDistance) || ...
316                     (nearestDistance == hittingDistance)
317                 % The walker is close enough to the nearest particle in the
318                 % cluster that there is a chance of collision at the next
319                 % step. We extract the sufficiently close particles from
320                 % the clusterMatrix.
321                 xround = middle + round(x);
322                 yround = middle - round(y);
323                 collisionCandidateMatrix = clusterMatrix(round(yround...
324                     - centreDistance):round(yround + centreDistance),...
325                     round(xround-centreDistance):round(xround...
326                     + centreDistance));
327
328                 % This matrix contains the labels of stuck particles near
329                 % the walker. We can now extract their labels.
330                 collisionCandidates = [];
331                 for i = 1:2*centreDistance+1
332                     for j = 1:2*centreDistance+1
333                         if collisionCandidateMatrix(j,i) ~= 0
334                             collisionCandidates = [collisionCandidates...
335                                 collisionCandidateMatrix(j,i)];
336                         end
337                     end
338                 end
339
340                 % Now we have the labels we can extract the exact
341                 % co-ordinates from stuckParticles. We have the current
342                 % position of the walker (x,y) as well as the positions of
343                 % all the candidates. We check each candidate to see if
344                 % there is a collision using our quadratic equation.
345
346                 % Assume we can take a step of minimum size and if there is
347                 % collision we will take a smaller step.
348                 stepSize = minStepSize;
349
350                 for i = 1:length(collisionCandidates)
351                     % Extract details of each individual potential particle
352                     % collision.
353                     tempLabel = collisionCandidates(i);
354                     xcol = stuckParticles(tempLabel,1);
355                     ycol = stuckParticles(tempLabel,2);
356                     rcol = stuckParticles(tempLabel,3);
357                     % Distance between particle centres.
358                     dparticles = rcol + rparticle;
359
360                     % Solve quadratic equation.
361                     A = 1;
362                     B = 2*((sin(randAngle))*(x - xcol) +...
363                         (cos(randAngle))*(y - ycol));
364                     C = (xcol - x)^2 + (ycol - y)^2 - dparticles^2;
365                     coefficients = [A,B,C];
366                     equationRoots = roots(coefficients);
367
```

```
368                     % If there exists a real, positive solution less than
369                     % minStepSize we have collision.
370                     for j = 1:2
371                         root = equationRoots(j);
372                         if (imag(root) == 0) && (root > 0)
373                             if (root < minStepSize) ||...
374                                     (root == minStepSize)
375                                 % Then we have a collision with the
376                                 % particle after a root step, so adjust the
377                                 % stepSize. This ensures if there are
378                                 % multiple collisions we take the smallest
379                                 % step size.
380                                 if root < stepSize
381                                     stepSize = root;
382                                     % Change bool to exit loop.
383                                     aggregate = 1;
384
385                                 end
386                             end
387                         end
388                     end
389                 end
390
391             % Else we are not near enough to aggregate but are closer
392             % than the maxInspectedDistance.
393             elseif (nearestDistance > hittingDistance) &&...
394                     (nearestDistance < maxInspectedDistance)
395
396                 stepSize = ...
397                     max(nearestDistance - hittingDistance,minStepSize);
398
399             % Else we are at least as far as the maxInspectedDistance
400             % from the cluster.
401             elseif nearestDistance == maxInspectedDistance
402
403                 stepSize = max(distanceFromCenter - maximumDistance...
404                     - hittingDistance,maxInspectedDistance...
405                     - hittingDistance);
406
407             end
408
409             % Now we can take the step with the appropriately calculated
410             % stepSize.
411             x = x + stepSize*sin(randAngle);
412             y = y + stepSize*cos(randAngle);
413         end
414
415     end
416
417     % If the particle has aggregated then we need to update all our
418     % matrices.
419     if aggregate
420         particleNumber = particleNumber +1;
421         % Update stuckParticles.
```

```
422         stuckParticles(particleNumber,1) = x;
423         stuckParticles(particleNumber,2) = y;
424         stuckParticles(particleNumber,3) = rparticle;
425
426         % Update the clusterMatrix.
427         ymat = middle - round(y);
428         xmat = middle + round(x);
429         clusterMatrix(ymat,xmat) = particleNumber;
430
431         % Now we have updated the stuckParticle matrix and clusterGrid we
432         % need to update our distanceGrid with the new distances. So we
433         % take a submatrix around our new aggregated particle and compare
434         % to the vicnityMatrix corresponding to the rounded particle radius
435         % of the aggregated particle.
436
437         roundedParticleRadius = round(rparticle);
438         % Select appropriate Vicinity Matrix.
439         tempVicinityMatrix = vicinityMatrices{roundedParticleRadius};
440         submatrixSize = 2*maxInspectedDistance +...
441             2*roundedParticleRadius +1;
442         % Take a submatrix around the close enough values in the
443         % distanceMatrix.
444         submatrix = distanceMatrix(ymat - maxInspectedDistance -...
445             roundedParticleRadius:ymat + maxInspectedDistance +...
446             roundedParticleRadius,xmat - maxInspectedDistance -...
447             roundedParticleRadius:xmat + maxInspectedDistance +...
448             roundedParticleRadius);
449         % Update each entry in this submatrix by taking the least value
450         % between the current cell value and the vicinityMatrix.
451         for i = 1:submatrixSize
452             for j=1:submatrixSize
453                 submatrix(j,i) = ...
454                     min(tempVicinityMatrix(j,i),submatrix(j,i));
455             end
456         end
457         % Put the updated submatrix back in.
458         distanceMatrix(ymat - maxInspectedDistance -...
459             roundedParticleRadius:ymat + maxInspectedDistance +...
460             roundedParticleRadius,xmat - maxInspectedDistance -...
461             roundedParticleRadius:xmat + maxInspectedDistance +...
462             roundedParticleRadius) = submatrix;
463
464         % Now we update the radii attractionMatrix. Define the maximum
465         % attraction distance.
466         tempAttractionDistance = ...
467             round(unitAttractionDistance*rparticle*maxParticleRadius);
468         radiiNeededSubmatrixSize = 2*tempAttractionDistance + 1;
469         % Take a submatrix of the close enough values to update.
470         radiiNeededSubmatrix = attractionMatrix(ymat - ...
471             tempAttractionDistance:ymat + tempAttractionDistance,...
472             xmat - tempAttractionDistance:xmat + tempAttractionDistance);
473         % Define middle of this submatrix.
474         radiiNeededMiddle = ...
475             unitAttractionDistance*rparticle*maxParticleRadius;
```

```
476         % For each cell calculate the radii needed for a walker to feel
477         % attraction from the newly aggregated particle. Update the entry
478         % of the cell to the minimum of the old cell value and this newly
479         % calculated radius.
480         for k = 1:radiiNeededSubmatrixSize
481             for l = 1:radiiNeededSubmatrixSize
482                 xdif = abs(k - radiiNeededMiddle);
483                 ydif = abs(l - radiiNeededMiddle);
484                 distanceFromMiddle = sqrt(xdif^2 + ydif^2);
485                 % Calculate radii needed.
486                 radiiNeeded = ...
487                     distanceFromMiddle/(unitAttractionDistance*rparticle);
488                 % Update if this radii is smaller than previously.
489                 radiiNeededSubmatrix(l,k) = ...
490                     min(radiiNeededSubmatrix(l,k),radiiNeeded);
491             end
492         end
493         % Put the updated submatrix back in.
494         attractionMatrix(ymat - tempAttractionDistance:ymat +...
495             tempAttractionDistance,xmat - tempAttractionDistance:xmat +...
496             tempAttractionDistance) = radiiNeededSubmatrix;
497 
498         % Check if we need to update the radius of the cluster. Calculate
499         % how far the newly aggregated particle is from the seed.
500         distanceFromCenter = sqrt(x^2 + y^2);
501         % If this distance is greater than the previous radius, update to
502         % this new radius.
503         if distanceFromCenter > maximumDistance
504             maximumDistance = distanceFromCenter;
505         end
506         % Display the particle number aggregated.
507         disp(['particle aggregated: ' num2str(particleNumber)])
508 
509         % End of script check.
510         if particleNumber == numberOfParticles
511             endscript = 1;
512         end
513     end
514 end
515 
516 %% Plot and Manipulate Results
517 
518 % Now we have aggregated all the particles and are storing the exact
519 % locations of their centres and their radii in stuckParticles. We now plot
520 % the particles and perform some calculations. Define some variables for
521 % plotting.
522 ang=0:0.01:2*pi;
523 particleCosine = cos(ang);
524 particleSine = sin(ang);
525 clf
526 clusterMass = 0;
527 
528 % For each aggregated particle we plot it with the appropriate radius and
529 % add its mass to the clusterMass.
```

```
for i = 1:numberOfParticles
    xcent = stuckParticles(i,1);
    ycent = stuckParticles(i,2);
    rcent = stuckParticles(i,3);
    clusterMass = clusterMass + rcent^2;
    hold on
    plot(xcent+rcent*particleCosine,ycent+rcent*particleSine,'r');
end

% Now we find appropriate axes limits.
miny = min(stuckParticles(:,1));
maxy = max(stuckParticles(:,1));
minx = min(stuckParticles(:,2));
maxx = max(stuckParticles(:,2));
xdif = maxx-minx;
ydif = maxy-miny;
maxaxis = max(xdif,ydif)/2;
xlim([- maxaxis - 10,maxaxis + 10])
ylim([- maxaxis - 10,maxaxis + 10])
hold off

% Output the desired values.
radius = maximumDistance
clusterMass
timeElapsed = toc(startTime)

% Finally we save the stuckParticles matrix to save our cluster for future
% analysis we may wish to perform.
save('expdistattraction.mat','stuckParticles');
```