PHYS 389 Computer Modelling Project 2020-2021

Simulating 2D Diffusion-Limited Aggregation and Brownian Tree Formation

Melissa Chung

BSc (Hons) Physics

Department of Physics, Lancaster University, United Kingdom

Dated: 29th March 2021

Abstract

A computer simulation and analysis for the fractal growth of diffusion limited aggregation (DLA) clusters in two-dimensional Euclidean space is presented. DLA is an irreversible kinetic process whereby particles undergoing persistent random walks aggregate around a fixed seed, generating fractal structures known as Brownian trees. Its computer implementation was conducted in Python, using the *pygame* set of modules for simulation. First simulating 1D, 2D and 3D simple random walks of constant step size on a discrete square lattice, average displacement and root-mean-square displacement were calculated. These results were compared alongside known, well-defined values and verified the accuracy of the simulation. An extension to many particles undergoing 2D random motion around a seed point successfully modelled DLA. The calculation of Hausdorff fractal dimension D_h for a 2D DLA cluster presented a firm agreement of $D_h = 1.7$ with existing literature. Introducing different combinations of domain shapes and attractor geometries presented a final value of $D_h = 1.74$ for a circular spawn shape and $D_h = 1.69$ for a square spawn.

Physics | Lancaster University

Contents

I. Introduction

First proposed by Witten & Sander in 1981 [1], the study of diffusion-limited aggregation (DLA) by computer simulation was seminal for enhancing the understanding of structures produced by irreversible growth processes. The fundamental basis of the model involves a particle being introduced into a system at a random position, before undergoing a random walk until it encounters an existing DLA structure. This will initially be a single stationary 'seed' particle, but upon contact the particle will permanently adhere to the seed, becoming part of the DLA structure. The growing aggregate results in the formation of so-called 'Brownian trees' which exhibit characteristic fractal geometries, a concept first introduced by Mandelbrot [2]. Practical applications of DLA are found throughout the natural world, from electrochemical deposition [3] to tumour growth [4] and snowflake formation [5].

To begin, a model of simple random walks of constant step size is developed, to represent the stochastic (random) motion of small particles in discrete time and space. This is conducted in 1D, 2D and 3D Euclidean space, and then extended to variable step size using a pseudo-random number generator. Simulation results are however restricted by the discrete nature of the computer, so continuous Brownian motion cannot be realistically modelled. From this simple model, a more complex DLA simulation is built by introducing a seed and initiating particle motion from a defined domain boundary. The consequent adhesion of particles and growth of a fractal structure is characterised by the Hausdorff dimension D_h [6].

Analysis of discrete random walk accuracy takes the form of verifying the average and root-mean-square (rms) displacements against expected values, based on statistical calculations. This is carried out for random walks in 1D, 2D and 3D, across a range of step numbers and walk iterations. An investigation into the generated DLA structures focuses on calculating the Hausdorff dimension D_h for different particle spawn and seed shapes. This non-integer value is fundamentally a measure of a fractal's 'roughness', and exists as a power law relation between mass and radius of the crystal.

A collection of Python packages and modules are utilised to generate, visualise and analyse these stochastic processes. Composition is used in the main DLA simulation to draw upon the merits of object-oriented programming, and units tests are implemented to verify functionality of the program. Storage and visualisation of data is optimised in *pandas* DataFrames, and animations are used to illustrate the unpredictable motion of many particles.

II. Theoretical Background

A. Random Walks

In 1827, Robert Brown described the erratic and irregular movement of pollen grains in water [7]. The eponymous phenomenon of Brownian motion details a stochastic (random) process of particles colliding in a fluid, whereby the individual motions of each particle may be described by a continuous random walk. A random walk consists of a succession of random steps, which are statistically independent and typically unbiased in direction. It is assumed in 2D and 3D that motion in each random variable (e.g. x, y, z-directions) is independent and identically distributed.

Two key analytical parameters for random walks of constant step size are the average displacement $\langle d \rangle$ and root-mean-squared displacement $\sqrt{\langle d^2 \rangle}$. As indicated in Equations 1 & 2,

$$\langle d \rangle = \sum_{i=1}^{N} \langle d_i \rangle = \sum_{i=1}^{N} [0.25 \times (-1) + 0.25 \times 1] = 0 \tag{1}$$

$$\langle d^2 \rangle = \langle x^2 + y^2 \rangle = N \langle x_i^2 \rangle = N \tag{2}$$

The average displacement should equal zero, which physically means that on average, particles will go nowhere.

The rms displacement gives an indication of how much the particles spread, or in other words the standard deviation of the normal distribution. It is equivalent to the square root of N (number of steps), assuming unit step size.

B. Pseudo-Random Number Generators

Computers, being deterministic machines in nature, cannot generate purely random sequences of numbers. Instead, particular algorithms are used to compute sets of numbers which approximate statistically random characteristics, but are not intrinsically random. These are called pseudo-random number generators. Each number in a sequence is generated from the previous one based on a well-defined algorithm(, such as the linear congruential method).

The pseudo-random number generator used in this model is taken from the Python module *numpy.random*. In particular, the function *numpy.random.randn* generates a random number which is normally distributed under mean $\mu = 0$ and variance $\sigma^2 = 1$. This uses a random seed to initialise the generator and fully determine the number sequence. By keeping the same seed value, the reproducibility of results can be maintained.

C. Hausdorff Dimension

Fractal geometries can be characterised by the non-integral Hausdorff dimension D_h.

$$M(R) = R^{D_h}$$

where $M(R)$ is the mass of the cluster, dependent on its radius of gyration R. Since there is a power-law dependence between $M(R)$ and R, a log-log plot of $\ln(M(R))$ against $\ln(R)$ should yield a straight line of slope coefficient D_h. This is the method by which the Hausdorff dimension will be calculated.

A line (1D) has D=1, whilst a plane (2D) has D=2. Hence higher D_h value give rise to a more compact and dense structure, whilst a lower value yields a more tenuous and open structure.

The fractal dimension D, is related to the exponent a by D, = d - alpha, where d is the dimension of the embedding space. All simulations are performed on normal Euclidean lattices where the dimensionality is an integer.

III. Technical Implementation

The DLA kinetic growth model relies on the diffusion of particles through a medium, limited by a fixed seed to which particles then aggregate to form fractal clusters. In the original model, a seed particle is first placed at the centre of a square lattice, before a second particle is diffused from a randomly selected spawn site, and then adheres (or aggregates) to the seed particle. A two-particle cluster is hence formed when the second particle travels to any site adjacent to the seed. This process repeats again for further randomly generated particles, to eventually form a large DLA aggregate (or Brownian tree). The simplest and most effective square lattice was considered to be the pixel grid of a screen.

Computational implementation of DLA limited the simulation of continuous random walks to discrete space, since individual particles were represented by single pixels on a screen. The finite length of time available further imposed restrictions on simulation complexity, together with the computational resource available.

As a result, numerous simplifications were made to the model for both efficiency and ease, without diminishing accuracy of results to a severe extent. The most notable change was the shift to discrete steps of fixed step size, which allowed direct implementation of random walks on the square pixel lattice. A 'spawn shape' parameter was introduced to limit the distance required by any spawning particle to reach the DLA cluster, and hence the time taken for the simulation to complete. This adjustment bore no effect on the properties of the fractal cluster produced, and only served to improve computational time. A 'wrap around' operation was also performed to ensure particles did not leave the domain bounds once near, and particles were 'recycled' from the spawn once aggregated to further improve performance.

Further, an approximation to physical systems was considered to be reasonable when the density of particles was low enough such that they could be modelled independently. Consequently, it was assumed that the probability of interaction between unaggregated particles was negligible.

Modifications to this simple model permitted flexibility and assessment of parameter changing. The most obvious parameter to vary was the number of particles spawned at a time, with more particles increasing the simulation time at the expense of computational power. In addition, a set of 'attractor geometries' representing seed shape was created, extending the central seed beyond the typical single particle, to a line, circle, ellipse or square. The 'spawn shape' parameter, allowing either a square or circular spawn, provided an investigation into biased fractal growth since square spawns preferentially aggregated particles at certain angles. The sizes in pixels of both the seed and spawn shapes were adjustable and thus open to exploration.

IV. Program Design

Program design encapsulated the file and class structure used. This was intended to organise the code into clear, unambiguous components in order to optimise maintenance and reproducibility. The key non-standard Python packages and modules applied were *random*, *pandas*, *pygame*, *matplotlib.animation* and *pytest*. Class structure was formed within the main DLA file *dla_simulation.py*, between two classes (*Particle* and *Application*) via composition.

Across the simple random walk and DLA models, there were 4 files composing the essential classes for successful simulation and analysis. In order of development, these files were:

constantstep.py: Generated and plotted random walks in 1D, 2D and 3D with fixed step size, using *random.choice*. Calculated average and rms displacements over many walk iterations, then plotted distance of a particle as a function of steps taken away from the start point. All plots were created using *matplotlib*.

variablestep.py: Extended the 1D, 2D and 3D random walks of *constantstep.py* to variable step size and multiple particles. Used *numpy.random.randn* to vary step size based on the standard Gaussian distribution, and display walks for multiple particles on the same plot. Average displacement over many walk iterations was calculated. Animations were also created for multiple particles at the same time, using *matplotlib.animation*. Positional and time data was stored in a *pandas* DataFrame, for ease of plotting and animation.

dla_simulation.py: Simulated the formation of a Brownian tree by DLA in 2D. Used *pygame* to implement an animation of particle aggregation to a central seed, as described in Section III. Usage of composition allowed a class hierarchy to form, with a composite class *Application* and a component class *Particle*. This allowed implementation of many-particle trajectories simultaneously, through instantiation of the *Particle* class within a loop. The main *Application* class initialised all relevant attributes, before creating a seed under a determined spawn shape, and then updating the positions of n particles sequentially. The direction of each step increment was implemented using *random.choice*, with no bias applied in any one direction. The DLA clusters freely grew until they reached a specified size limit, upon which the simulation ended.

frac_dim.py: Calculated the Hausdorff dimension D_h for a 2D DLA cluster, using Equation x. Extended to iterate calculations over many DLA cluster radii, allowing a more accurate D_h value to be obtained. Used *matplotlib* to generate plots of $\log(\text{mass})$ against $\log(\text{radius})$ with best fit straight lines, to visualise the distribution of data points across a wide range of mass and radius values. Data for the calculations and plots was stored and saved in *pandas* DataFrames.

There were also 4 test files containing unit tests corresponding to each file using *pytest*. The specifics of these tests will be further outlined in Section V.

The procedural flow of *dla_ simulation.py* was significantly more complex than the other files, and thus a flow chart was created to represent its logic visually and unambiguously. This is included in Figure 1.

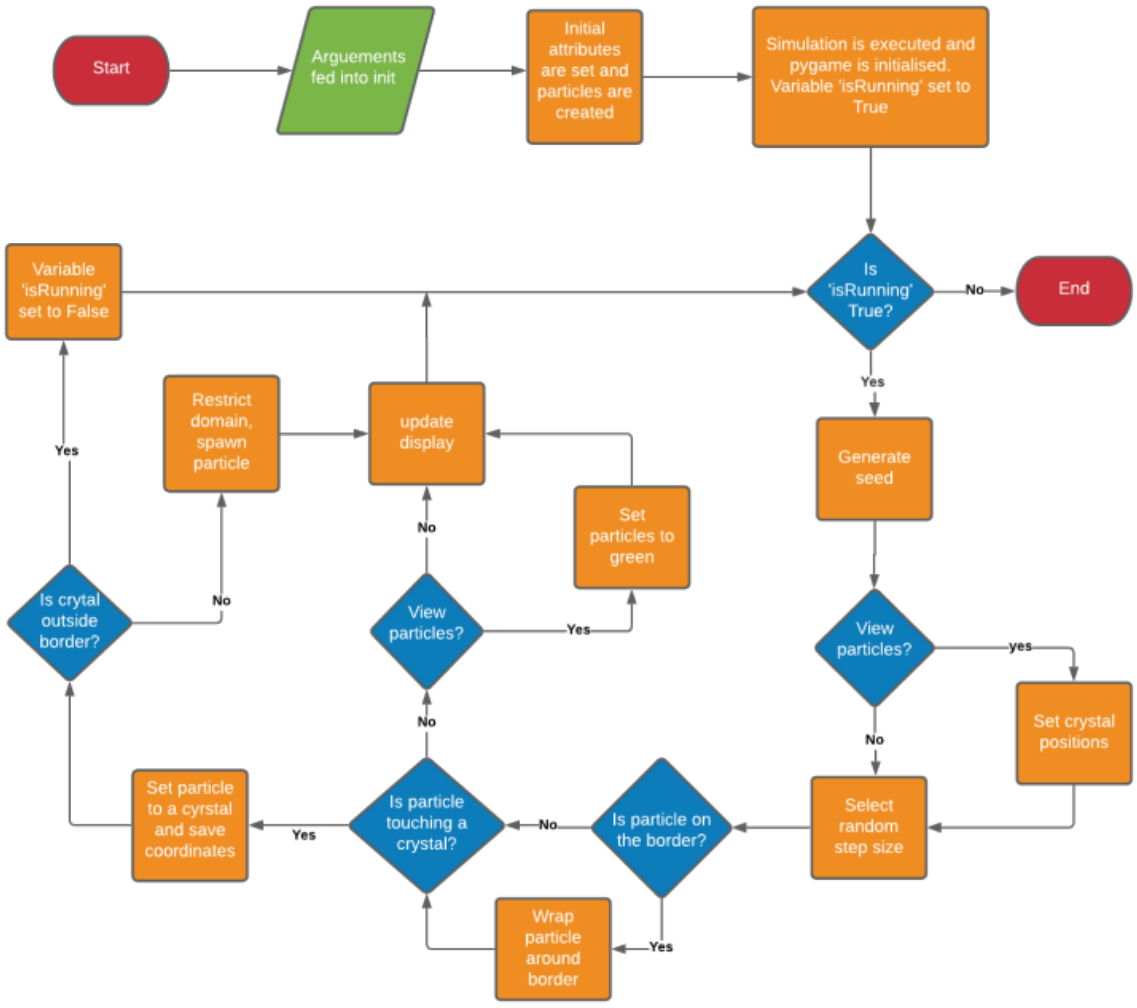


Figure 1: A flowchart for the main DLA simulation file.

V. Testing

Testing of the functions and classes created was conducted using *pytest*, a framework developed for unit testing in Python. This process was paramount for code development, since it ensured the code operated as expected by easily identifying errors both trivial and fatal in nature.

Separate files containing test functions were created to correspond to each tested file. These test files and function were named with a prefix *test_* $\{$*file*/*function*$\}$ for clarity. *pytest* fixture decorators were introduced to define input parameters for initialising test functions, providing an effective alternative to repeatedly instantiating an object externally to each function. Another *pytest* decorator, parametrize, was used to allow passing of multiple input parameters into a single test function. This also removed redundancies and increased test file succinctness.

Due to the stochastic nature of random walk and DLA implementation, it was not possible to conduct tests without preserving reproducibility of results. A random seed state (unrelated to the DLA fixed seed particle) was defined to initialise the pseudo-random number generator used. As described in Section IIB, using the same random seed state produced the exact same set of randomly-distributed results. This allowed precise testing of expected output values for particular inputs, using assert statements. Any raised exceptions in the code were also tested using *pytest.raises*. Table 1 outlines the key tests considered essential for successful running of the model, with the input and output values obtained.

Test Function Name	**Key tests**	**Expected Value**	**Obtained Value**	**Passed**
test_constant_step_init()	Number of steps, N	10	10	✓
	Step size, ss	2	2	✓
	Iterations	5	5	✓
test_gen_random_walk()	Positional co-ordinates in 1D, 2D, 3D	-2, (-4, -2), (-6, -4, 4)	-2, (-4, -2), (-6, -4, 4)	✓
test_calc_displacements()	Average and rms displacements in 2D	(3.6, 6.511528238439882)	(3.6, 6.511528238439882)	✓
test_variable_step_init()	Total time, T	100	100	✓
	Number of walkers, M	3	3	✓
	Time interval, dt	5	5	✓
test_calc_displacements()	Average displacement in 2D	2.1	2.1	✓
test_particle_init()	Initial x position of particle	5	5	✓
	Initial y position of particle	10	10	✓
test_particle_update()	Updated x position of particle	1	1	✓
	Updated y position of particle	2	2	✓
test_application_init()	Application window size	(800, 600)	(800, 600)	✓
	Total number of particles	100	100	✓
	Seed shape	'line'	'line'	✓
	Spawn shape	'square'	'square'	✓
	Padsize (size of square spawn)	70	70	✓
	Radius (size of circle spawn)	70	70	✓
	Crystal size limit	100	100	✓
test_gen_seed()	Exception is raised	Error caught	Error caught	✓

Table 1: A table listing major test functions with their key tests, along with expected and obtained values and a pass or fail mark.

VI. Results & Analysis

A. Simple Random Walks

To begin, random walks with constant step size were generated across a range of total number of steps, in 1D, 2D and 3D. Position plots were then created, along with plots of variation of total distance against number of steps. The latter set of plots are shown in Figures 2, 3 & 4.

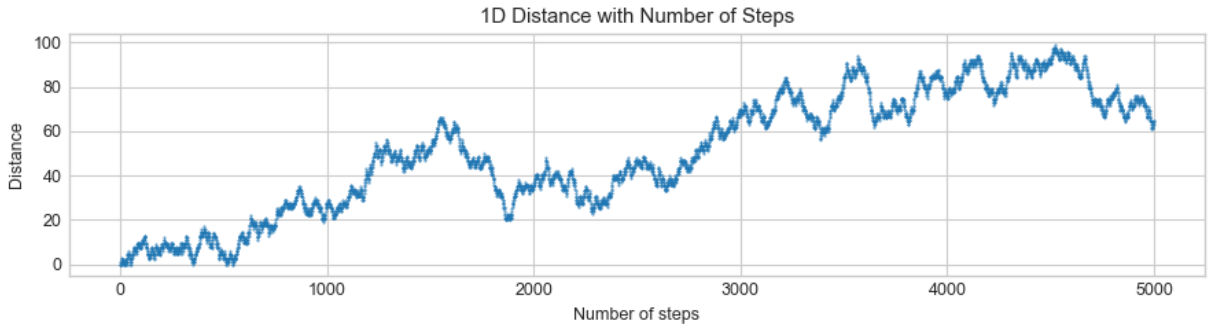


Figure 2: 1D random walk, showing distance as a function of number of steps, over 5000 steps.

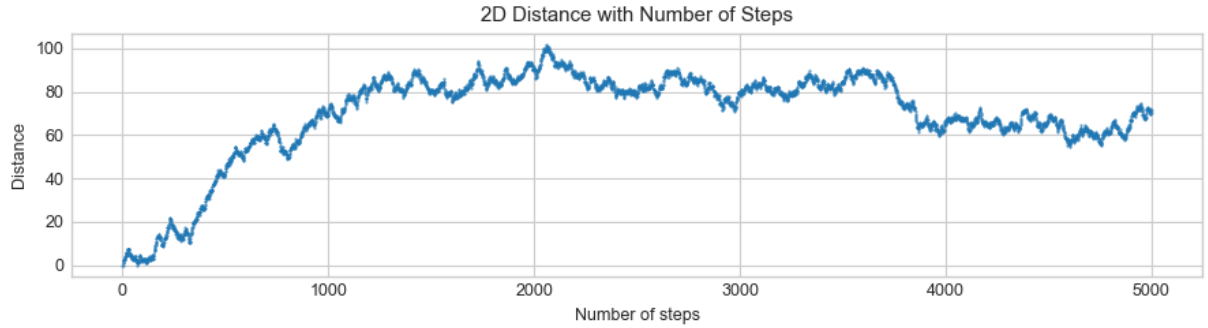


Figure 3: 2D random walk, showing distance as a function of number of steps, over 5000 steps.

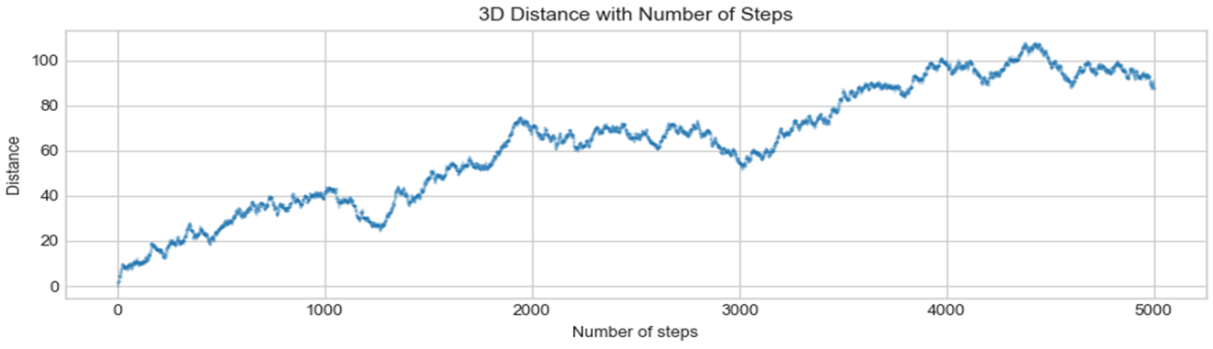


Figure 4: 3D random walk, showing distance as a function of number of steps, over 5000 steps.

As depicted, there is a consistent growth of distance from the starting point with number of steps, or time. Since these plots are results for a single particle over one iteration, being randomly

generated they are subject to significant change between runs. However, it is typical from this model that a particle will tend to move further away from its starting point as the system evolves with time. There appears to be no significant variation between the plots for 1D, 2D and 3D.

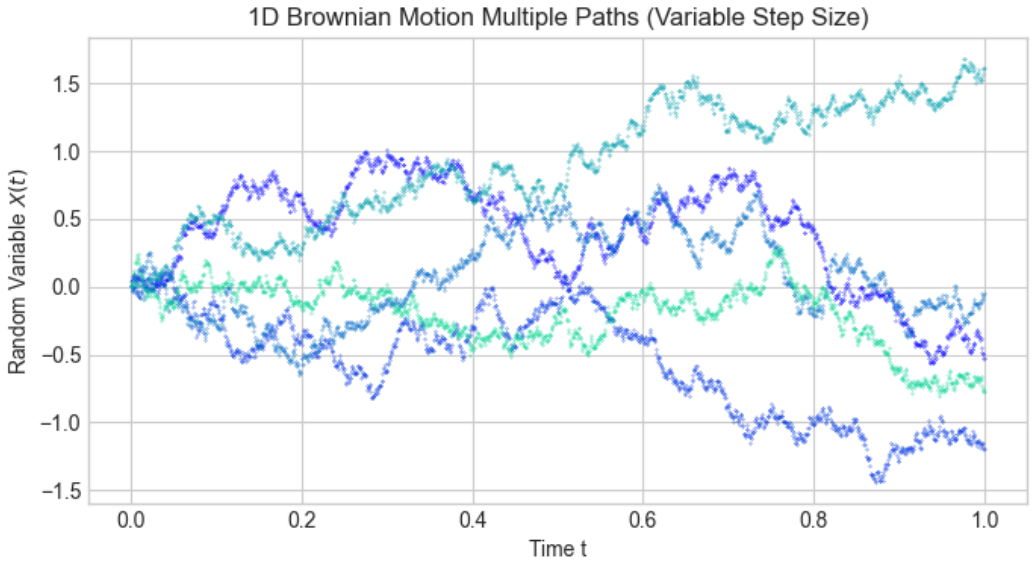


Figure 5: 1D random walks of 5 particles taking 1000 steps of variable length over 1 second.

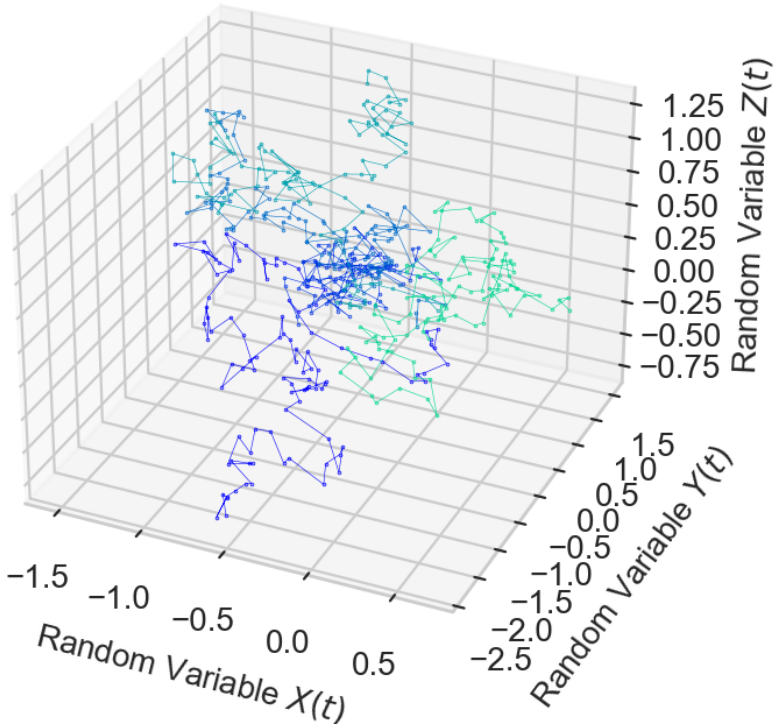


Figure 6: 3D random walks of 5 particles taking 100 steps of variable length over 1 second.

An extension to variable step size was taken to mimic natural physical systems, and plots were produced for multiple random 'walkers' on the same axes. The plots in 1D and 3D are shown in Figures 5 and 6 respectively. Animations for variable step size were also generated to show the evolution of multiple particle with time, and qualitatively verify the functionality of the random walk results. Displaying largely similar visual results compared with constant step size, there appeared to be no distinct differences in the volatility of behaviour.

I. Average Displacement

In order to quantitatively determine the accuracy of the random walks generated, calculations were conducted for the average displacement $\langle d \rangle$ achieved by a particle over many iterations. As outlined in Section IIA, the expected value was zero. To increase accuracy, each average displacement was calculated for 1000 iterations, and then averages again over 50 runs. These values were selected to achieve more promising results whilst keeping computational power and time adequately low. The results across a range 500 to 5000 steps is presented in Table 2 for average displacements in 1D, 2D and 3D.

	Number of Steps	$\langle d \rangle$ in 1D for 50 runs	$\langle d \rangle$ in 2D for 50 runs	$\langle d \rangle$ in 3D for 50 runs
Constant Step	500	0.0146	-0.0770	-0.1044
	1000	-0.0248	0.0877	0.1344
	2500	0.2846	0.0913	-0.0854
	5000	-0.0570	0.3986	-0.15356
Variable Step	500	0.0368	-0.3972	0.2839
	1000	-0.9738	-0.4629	0.4820
	2500	0.1783	0.3860	-0.1782
	5000	0.3810	-0.0927	-0.2891

Table 2: A table displaying average displacement results in 1D, 2D and 3D across a range of step numbers. Results are displayed for constant step size and variable step size, and 1000 iterations are used per run. The average of the calculated $\langle d \rangle$ over 50 runs is presented.

Despite the number of runs and iterations, this data was still fairly unreliable, since repeat calculations yielded somewhat different results. This was likely due to the existence of a standard deviation range, which exists even for high iterations, albeit across a smaller range. There was no apparent dependence of $\langle d \rangle$ on the number of steps or spatial dimension, with results appearing to be truly random. As a result, it was difficult to deduce which of the constant step or variable step size methods produced better results. Overall however, no obtained $\langle d \rangle$ value exceeded ± 0.5, and thus all values were sufficiently close to zero.

II. Root-Mean-Squared Displacement

The root-mean-squared (rms) displacement $\sqrt{\langle d^2 \rangle}$ was restricted to calculation on random walks with a fixed step size. Directly as a result of the theory, the expected $\sqrt{\langle d^2 \rangle}$ value for a random walk of step number N and step size ss would be $ss\sqrt{N}$. Table 3 outlines the percentage difference between the obtained and expected $\sqrt{\langle d^2 \rangle}$ values across a step number range of 500 to 10000.

Number of Steps	% diff. $\sqrt{\langle d^2 \rangle}$ in 1D	% diff. $\sqrt{\langle d^2 \rangle}$ in 2D	% diff. $\sqrt{\langle d^2 \rangle}$ in 3D
500	1.790	1.028	3.166
1000	0.179	1.047	1.248
2500	2.233	1.025	0.142
5000	0.178	4.276	1.467
10000	2.291	0.466	0.857

Table 3: A table displaying rms displacement results in 1D, 2D and 3D across a range of step numbers from 500 to 10000. Results are displayed for constant step size random walks, and 1000 iterations are used per run.

Once again, there was no apparent relation between rms displacement and step number or dimension. The percentage errors were oddly high in some cases, but still sufficiently low to guarantee an accurate simulation.

Overall, the data from Tables 2 and 3 indicated a random variation of average and rms displacements with step number. In spite of a lack of dependence, it was clear from the proximity of $\langle d \rangle$ to zero, and the small percentage errors of $\sqrt{\langle d^2 \rangle}$ that this random walk simulation was sufficiently accurate.

B. 2D DLA Simulation

A core principle during software development of the simulation for diffusion limited aggregation was maintaining the ability to vary physical parameters. The four key parameters considered important for investigation were attractor geometry, particle spawn shape, sticking coefficient and the ability to view the motion of individual particles. Although irrelevant to the resulting structure of the DLA cluster, the 'viewing' option was considered useful both during code development and after for visual demonstration purposes. Figure 7 displays an image of the simulation when individual particle motion is selected for viewing, with each particle represented by a green pixel.

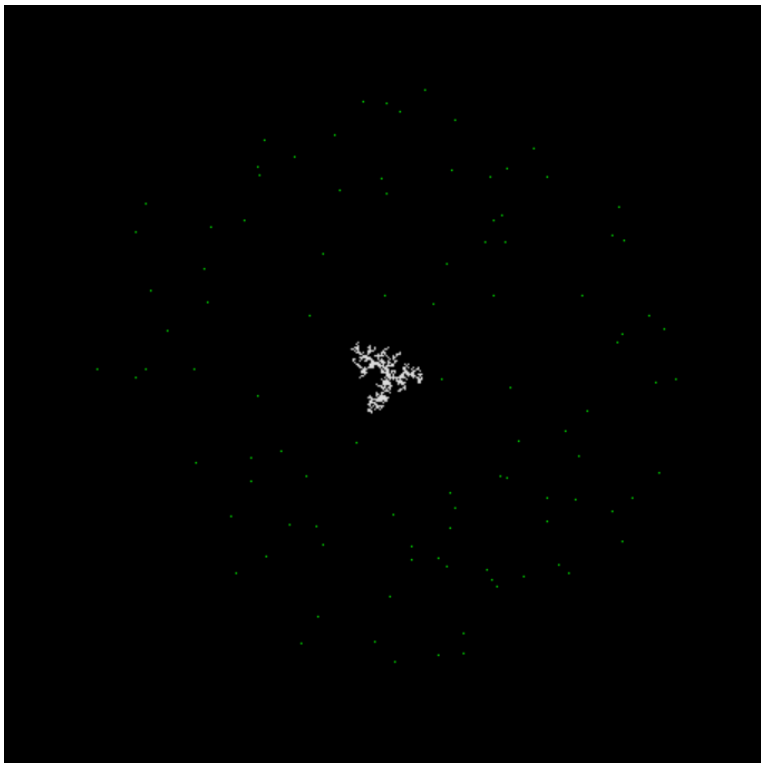
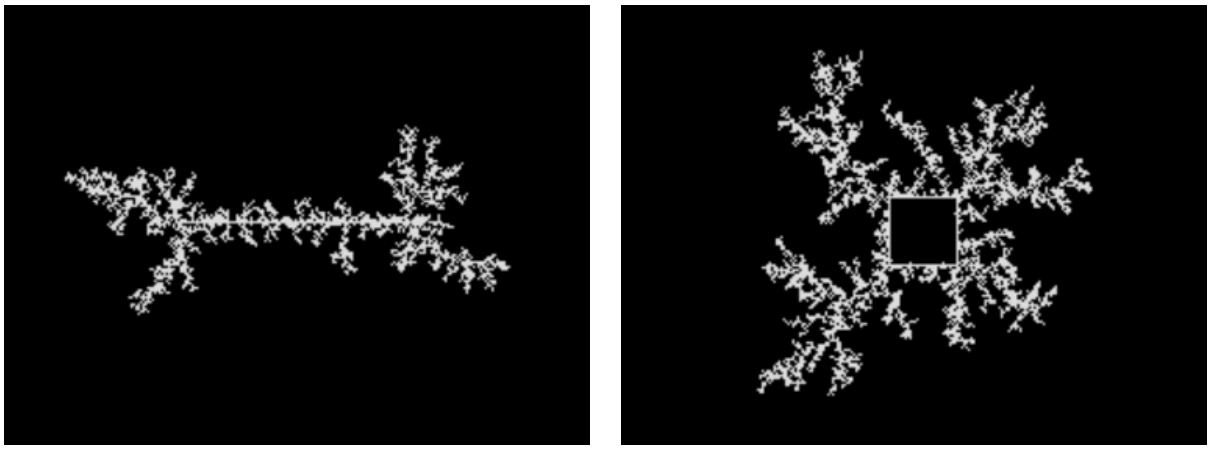


Figure 7: Capture of the DLA simulation upon selection of the viewing parameter for individual particle motion. Unaggregated particles were coloured as green pixels whilst the growing DLA cluster was coloured in white. A circular spawn and viewing domain was used.

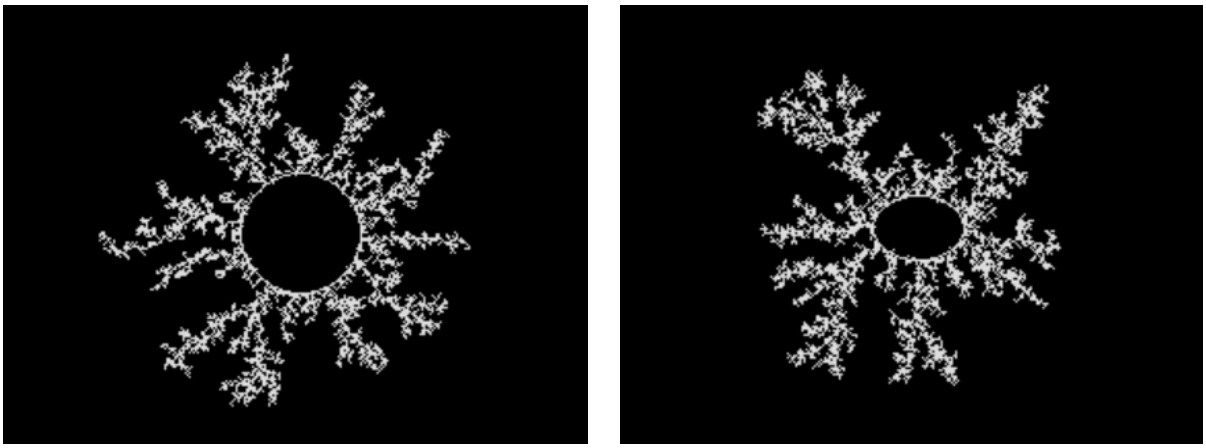
i. Attractor Geometry

Attractor geometry, or seed shape, was a significant determinant in DLA cluster formation structure. Typically a single particle seed, designated as a 'dot' seed shape, was used to create standard clusters. Thus although somewhat unphysical due to their uniformity, a line, square, circle and ellipse were chosen to investigate this effect. Results of the clusters formed under the same spawn shape, number of particles and sticking coefficient are included in Figures 8 and 9.



(a) DLA cluster with line seed (b) DLA cluster with square seed

Figure 8: DLA clusters formed using a circular spawn, based upon a line (a) and square (b) seed.



(a) DLA cluster with circular seed (b) DLA cluster with elliptical seed

Figure 9: DLA clusters formed using a circular spawn, based upon a circular (a) and elliptical (b) seed.

Under the diffusive regime of DLA, smooth growth was unstable due to the presence of noise, and as such any small bumps or protrusions tended to grow preferentially. The 'roughness' of these regions captured a greater proportion of the surrounding diffusive particles, and eventually developed into the tips of branches. Protruding branches prevented the inner sections of the clusters from capturing as many particles, further exacerbating this shielding effect. Such branches then developed bumps more due to roughness, growing new branches and propagating the dendritic structure observed. It was by this process that fractal growth, by definition of self-similarity, occurred.

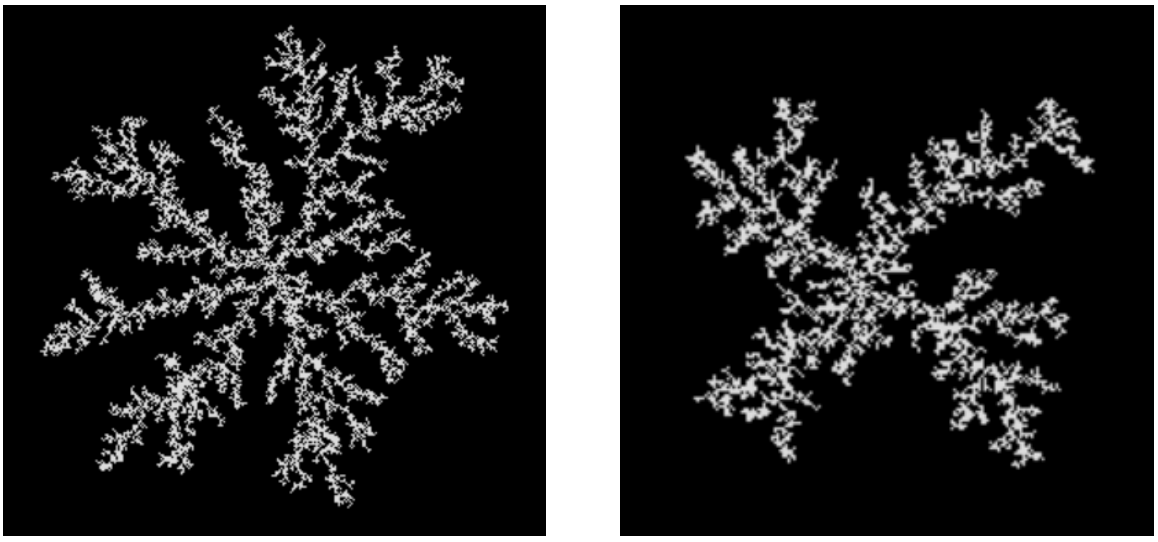
As a result of this unstable growth, any small fluctuation present initially would tend to become

larger and larger as the cluster grew. In an analogy with a physical phenomenon, the local electrostatic field at a tip or protrusion is higher than at the inner surface. As a result the probability of further growth is higher at the tip, proportionally to the strength of the electrostatic field. This occurrence is responsible for the dendritic growth of Lithium in rechargeable batteries, which results in short circuiting when dendrites pierce through the battery separator, and is hence a cause of major concern for safety.

A comparison of Figures 8 and 9 support the above theory, as the 'points' produced by the line ends and square corners acted as areas of irregularity. As a result, there was clear preferential cluster growth in these regions. Comparatively, the cluster uniformity was greatly improved for circular and elliptical seed shapes, with a barely perceptible effect of the sharpened 'corners' of the ellipse. Thus, it was clear from these results that a less sharp seed shape gave rise to a more uniform DLA cluster structure.

II. Spawn Shape

The next parameter of interest was the spawn shape, from which all particles initially originated before commencing a two-dimensional random walk. At first, by ease of implementation a square-shaped spawn was developed to contain the chosen seed shape. This was eventually extended to a circular spawn shape, with both shaped subject to variation in size. Figure 10 presents the typical DLA clusters formed from both spawn shapes, under the same seed shape and sticking coefficient.



(a) DLA cluster with circular spawn (b) DLA cluster with square spawn

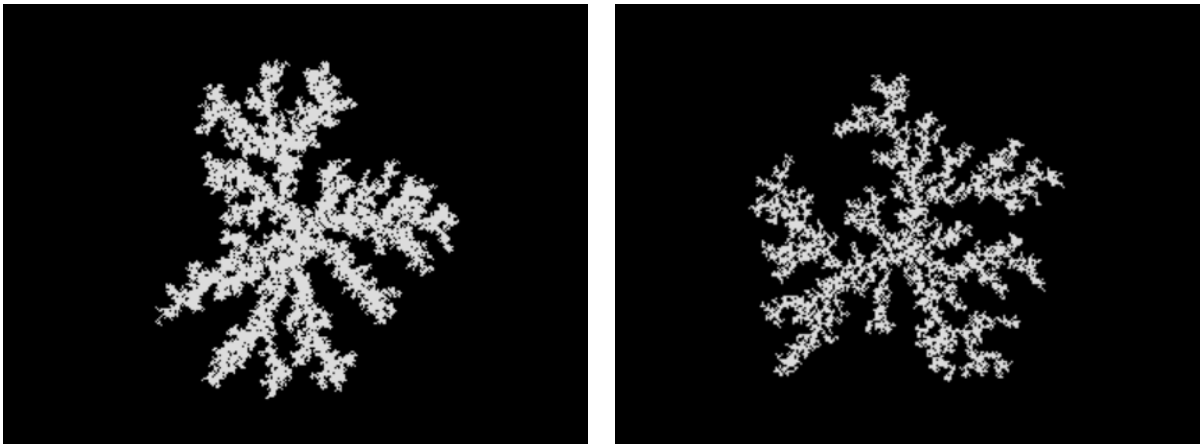
Figure 10: DLA Clusters formed upon a single 'dot' particle seed, using a circular (a) and square (b) surrounding spawn.

It was apparent by comparison of Figures 10a and 10b that a circular spawn gave rise to more uniform cluster growth in all directions, whereas the square spawn preferentially grew in particular directions. This discrepancy was attributed to the uniform distribution of particle spawning along the perimeter of both shapes, resulting in a statistically higher proportion of particles origination from the corner of the square. As a result, and as is evident, the square spawn cluster in Figure 10b grew largely in directions towards the corners of the square spawn. Thus, in order to preserve isotropic distribution and eliminate directional bias, a circular spawn was considered superior for garnering accurate results.

III. Sticking Coefficient

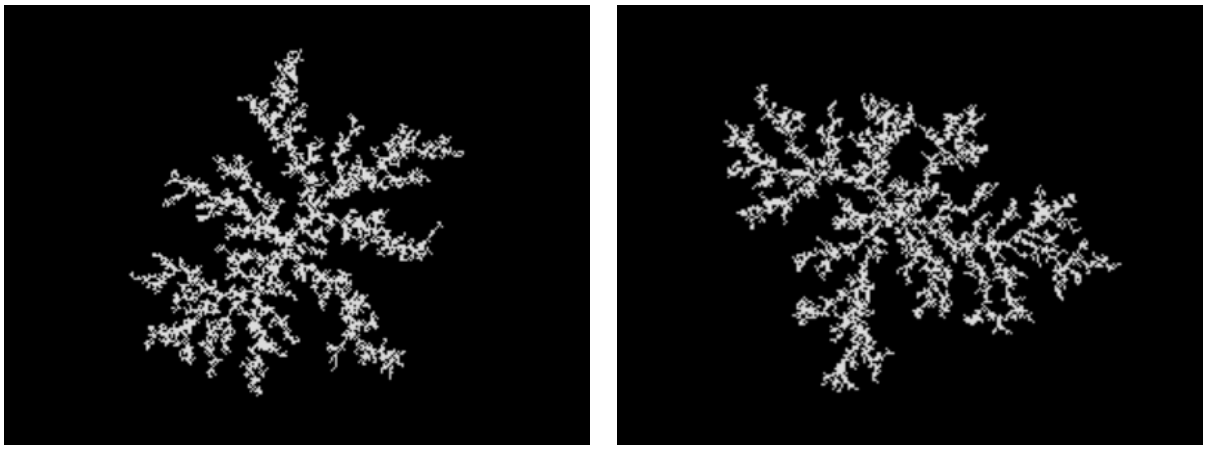
The introduction of a so-called 'sticking coefficient', denoted S, allowed the variation of aggregation probability on the cluster. Defined as the probability of a particle 'sticking' to the DLA structure, a sticking coefficient $S = 1$ would ensure aggregation upon contact, whereas a coefficient $S = 0$ would represent no cluster formation whatsoever. Hence, by varying the sticking coefficient, it could be expected that fractals of different Hausdorff dimension H_d could be observed.

This was indeed the case in Figures 11 & 12, which show DLA cluster formation at sticking coefficients of $S = 0.1$, $S = 0.4$, $S = 0.7$ and $S = 1.0$ respectively. As is visually evident, a lower sticking coefficient gave rise to a more dense structure, since more particles were given access to the inner structure of the cluster. One would thus expect a higher value for D_h due to a higher mass for the same radius.



(a) DLA cluster with $S = 0.1$ (b) DLA cluster with $S = 0.4$

Figure 11: DLA clusters formed using a circular spawn and single seed, of sticking coefficient $S = 0.1$ (a) and $S = 0.4$ (b).



(a) DLA cluster with $S = 0.7$ (b) DLA cluster with $S = 1.0$

Figure 12: DLA clusters formed using a circular spawn and single seed, of sticking coefficient $S = 0.7$ (a) and $S = 1.0$ (b).

IV. Hausdorff Dimension

The Hausdorff dimension D_h calculated using the circle spawn shape was $D_h = 1.74$. The Hausdorff dimension D_h calculated using the square spawn shape was $D_h = 1.69$. Compared with the accepted literature value of $D_h = 1.70$, these results were satisfactory and a reflection of the success of the project. The log-log plots of mass against radius for both of these are shown in Figures 13 & 14. The plotted values were calculated by analysing the cluster mass as a function of its radius $M(R) = R^{D_f}$, where $M(R)$ was the number of occupied sites (pixels) in the cluster R, and D_f was the fractal dimension of DLA-cluster.

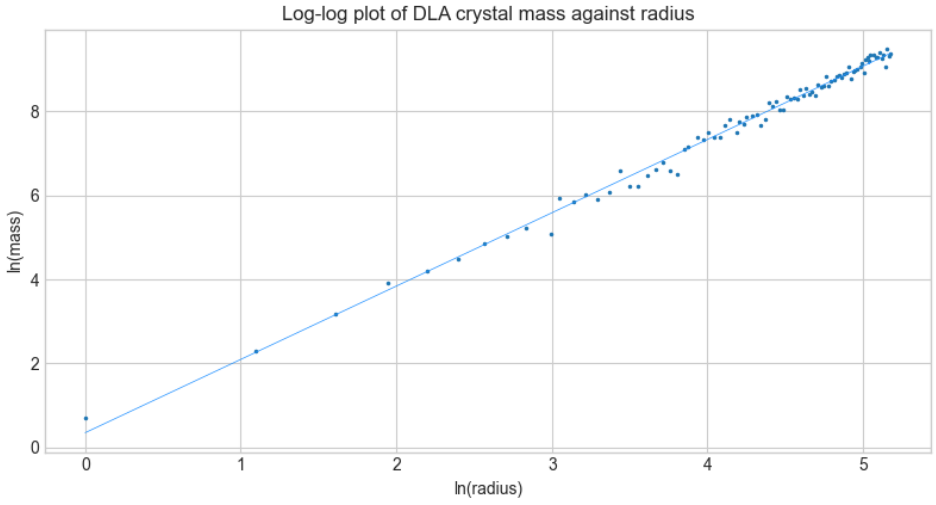


Figure 13: Log-log plot of mass against radius for a circular-shaped spawn.

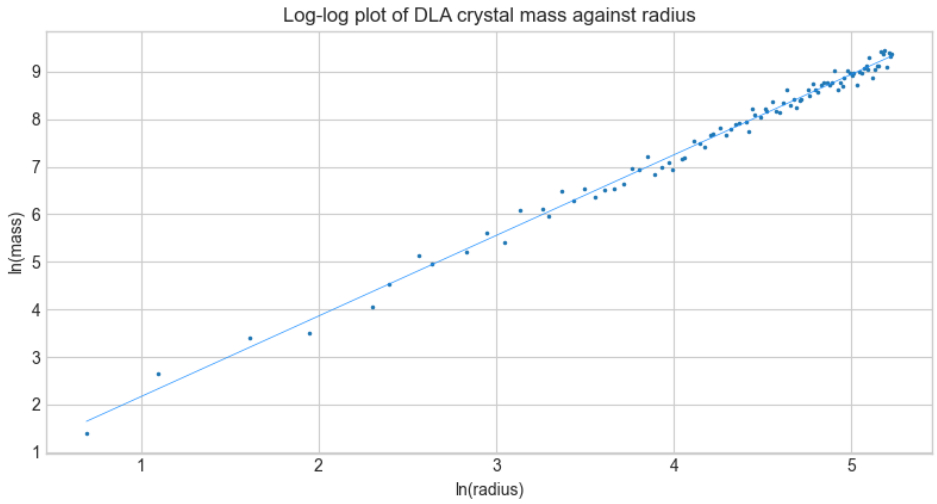


Figure 14: Log-log plot of mass against radius for a square-shaped spawn.

VII. Further Work

Despite obtaining satisfactory results for the displacement and Hausdorff dimension calculations, the final simulation presented some over-simplifications which limited its physical accuracy. The most apparent oversight was arguably the neglect of interaction between particles before aggregation. In the physical world, electrostatic forces of interaction exist between all particles subject to DLA, which would largely influence the path of a random walk. As a result, the exact DLA cluster shape formed using this simulation would not be equivalent to the 'real' one since different random walks would have been executed. Nevertheless, on a macroscopic scale results would remain representative, and thus the large increase in computational power required to track changes in all inter-particle forces would be unprofitable.

Another assumption was absence of a directional bias which varies in direction, time and bias strength. Considering a simple case of DLA-based frost formation, the variable force and direction of wind with time would have significant ramifications on final cluster properties. Simple implementation of a constant bias could first be executed by increasing of decreasing the probability of a particle moving in a certain direction. Then introducing a time-dependent bias, by either randomly varying the bias magnitude or defining some mathematical function, could result in the production of more 'natural' DLA clusters. A more effective implementation of variable bias could be rotation of the DLA structure during formation, with a change in the rate of rotation signifying a change in bias strength and direction [8]. Consequent analysis of fractals with these

additionally applied forces could give rise to results valuable to real-world applications.

A collection of further extensions to the simulation would allow deeper appraisal of its accuracy, and the ability to better model specific physical phenomena based on DLA. The most pronounced of these would be an extension to three dimensions, although this would require use of an alternative simulation library, since *pygame* does not support 3D capabilities. Introducing more than one particle size in a single simulation would allow representation of more complex systems, together with combining different seed shapes together. Adding the option to simulate rebound of particles off the system boundary, instead of wrapping around, would allow modelling of a closed system. Finally, a complex but justifiable shift from square lattice implementation to an off-lattice one would enable particles to take the shape of a circle and widen movement to any direction.

VIII. Conclusion

The time evolution of particles undergoing Diffusion Limited Aggregation (DLA) was successfully modelled in two dimensions by computer simulation. DLA was presented as a type of irreversible kinetic aggregation involving the formation of large fractal clusters from basic subunits undergoing diffusive random walks. In this simulation, unit particles were modelled on a discrete square lattice represented by pixels on a screen. Undergoing random walk trajectories, these would attach to a growing aggregate which began as a single fixed seed.

A basic simulation of random walks was first implemented using a pseudo-random number generator, with the calculated paths visualised through plots and animations. This was conducted in 1D, 2D and 3D, and the accuracy was evaluated by calculation of the characteristic average and rms displacements ($\langle d \rangle$ and $\sqrt{\langle d^2 \rangle}$). It was found that despite the random nature of the results obtained, they were sufficiently close to the expected values such that they verified an accurate random walk simulation. More specifically, the range of average displacements computed

Subsequently, the random walks of many particles in 2D was extended to DLA by introducing a fixed seed particle to which particles could adhere. This resulted in the growth of Brownian tree clusters demonstrating fractal branching properties. The fractal structure was found to vary with attractor geometry, spawn shape and sticking coefficient, with preferential growth on areas of high roughness. The Hausdorff fractal dimensions for a circular and square spawn respectively were found to be $D_h = 1.74$ and $D_h = 1.69$, which agreed closely with the widely accepted value of $D_h = 1.70$ for 2D DLA clusters [9] . It could thus be concluded that to a sufficient degree, with computational expenses in mind, the produced simulation provided a viable model for two dimension diffusion-limited aggregation.

REFERENCES

[1] T. A. WITTEN, L. M. SANDER (1981) *Author* Phys. Rev. Lett. 47 (19), pp. 1400-1403. doi:10.1103/PhysRevLett.47.1400 [Accessed 27-03-2021]

[2] B. B. MANDELBROT (1982) *The Fractal Geometry of Nature*, New York: W. H. Freeman and Company. [Accessed 27-03-2021]

[3] P. BOURKE (2006) *Constrained diffusion-limited aggregation in 3 dimensions*, Computers & Graphics, 30 (4), pp. 646-649. doi:10.1016/j.cag.2006.03.011 [Accessed 27-03-2021]

[4] I. M. JIANG, J. Y. KO, W. K. LIU, J. C. CHIANG (1994) *A Computer Simulation Study of Diffusion-Limited Aggregation on Sierpinski Lacunar Lattice*, Chinese Journal of Physics, 32. [Accessed 27-03-2021]

[5] C. D. WESTBROOK (2004) *Universality in snowflake formation*, PhD Thesis, University of Warwick, pp 7-10. [Accessed 27-03-2021]

[6] M. FERNÁNDEZ-MARTÍNEZ, M. A. SÁNCHEZ-GRANERO (2014) *Fractal dimension for fractal structures: A Hausdorff approach revisited*, Journal of Mathematical Analysis and Applications, 409 (1), pp.321-330. [Accessed 27-03-2021]

[7] S. BLUNDELL, K. BLUNDELL (2006) *Concepts in Thermal Physics*, Oxford: Oxford University Press, ch.19.4.

[8] A. LOSKUTOV, D. ANDRIEVSKY, V. IVANOV, K. VASILIEV, A.RYABOV (2000) *Fractal Growth of Rotating DLA-clusters* , Macromolecular Symposia, 160. arXiv:nlin/0004032 [Accessed 28-03-2021]

[9] P. MEAKIN (1983) *Diffusion-controlled cluster formation in 2—6-dimensional space*, Physical Review A, 27, pp.1495-1507.