# Diffusion Limited Aggregation

Sourabh Balgi

*M. Tech., Systems Engineering*

*Statistics and Machine Learning Lab*
*Computer Science and Automation*
Indian Institute of Science

*sourabhbalgi@gmail.com, sourabhbalgi@iisc.ac.in*

May 9, 2019

# Overview

# Fractals

- Fractals are irregular shapes which exhibits scale-invariance and self-similarity.
- Several real world structure such as mineral deposition, electrode deposition, Hele-Shaw flow, silicon wafer deposition, snow flake growth, coral formation, dielectric breakdown etc are some natural fractals.
- A fractal often has the following features:
  - It has a fine structure at arbitrarily small scales.
  - It is too irregular to be easily described in traditional Euclidean geometric language.
  - It is self-similar (at least approximately or stochastically).
  - It has a simple and recursive definition.

# Fractals : Classification

- Fractals can be classified according to their self-similarity:
  - **Exact self-similarity** is the strongest type of self-similarity. The fractal appears identical at all scales.
  - **Quasi self-similarity** is a loose form of self-similarity. The fractal appears approximately identical at different scales.
  - **Statistical self-similarity** is the weakest type of self-similarity. The fractal has numerical or statistical measures which are preserved across scales. All **DLA fractals** belong to this type.

(a) a (b) b

Figure: Examples of fractals. [a] Romanesco broccoli, showing self-similar form approximating a natural fractal. [b] High-voltage breakdown within a 4 in (100 mm) block of acrylic glass creates a fractal.

# Diffusion Limited Aggregation

- Diffusion-limited aggregation (DLA) is the process whereby particles undergoing a random walk due to Brownian motion cluster together to form aggregates of such particles.
- Proposed by T.A. Witten Jr. and L.M. Sander, DLA is applicable to aggregation in any system where diffusion is the primary mean of transport in the system.
- **Brownian trees** : Clusters formed due to fractal aggregate made by DLA.

# Brownian trees Formation

- Start with an empty lattice with only one seed particle at the center which is stuck.
- Start diffusing particles, one at a time, randomly along a given locus. The locus can be along the edges of a square or circumference of the circle etc.
- The diffused particle is allowed to undergo random walk ( Brownian motion ) until it encounters a neighbouring particle which is stuck and hence the current particle sticks as well.
- This process is continued until the diffusion locus is exhausted by the tree growth.

# Factors affecting Brownian tree formation

## Lattice Structure

The lattice selected for tree formation.
e.g. square lattice or circular lattice etc.

## Seed Particle Location

The location of the seed particle causes the change in the tree formation.
e.g. at the center, on the corners, on the edge etc.

## Diffusion Locus

The diffusion locus used for particle diffusion.
e.g. locus along the edges of a square lattice or along the circumference of a circular lattice etc.

# Factors affecting Brownian tree formation (continued ...)

## Stickiness (k)

The probability with which a particle is stuck on reaching a stuck neighbour.
e.g. $k = 1$ i.e. always sticks or $k = 0.5$ i.e. sticks only with a probability of 0.5 etc.

## Boundary Condition on the particles

The behavior of the particle on reaching the end of the lattice.
e.g. if the particles reaches the end of lattice, terminate and continue with a new random particle or bounce back into the lattice with an inverted velocity or reintroduce the same particle in a toroidally bound manner etc.

# Fractal Dimension

- Fractal Dimension ($D_f$) is a statistical quantity which indicates how completely the fractals fill the lattice space.
- If we take an object with linear size equal to 1 residing in Euclidean dimension $D$, and reduce its linear size by the factor $1/r$ in each spatial direction, it takes $N = r^D$ number of self similar objects to cover the original object. $r$ is called the characteristic linear dimension.

$$\log N = D \log r \tag{1}$$

$$D = \frac{\log N}{\log r} \tag{2}$$

$$D_f = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log \frac{1}{\epsilon}} \tag{3}$$

where $N(\epsilon)$ is the number of self-similar structures of linear size $\epsilon$ needed to cover the whole structure.

# Fractal Dimension (continued ...)

- Fractal Dimension ($D_f$) is very important in statistically inferring the attributes of the given DLA such as stickiness.
- As mentioned earlier, $D_f$ indicates how completely the lattice is filled i.e. $D_f$ is a function ($f$) of number of particles stuck in the lattice $N$.

$$D_f = f(N) \tag{4}$$

- $N$ in turn is a function ($g$) of stickiness ($k$) because stickiness determines the amount of particles getting stuck at any point in time.

$$N = g(k) \tag{5}$$

$$k = g^{-1}(N) \tag{6}$$

# Fractal Dimension (continued ...)

- For a given lattice with $N$ particles, we can then infer it's statistical attributes $D_f$ from Eq.(3), Eq.(4) and $k$ from Eq.(6).
- Specifically, Eq.(3) is what is referred as **Box Counting** method to predict the fractal dimension $D_f$ of a given fractal in the standard literature.

# Simulation parameters

- The main parameters for simulation are:
  - **Maximum lattice size (LS)** :
    In our experiments, we use a square lattice with side $LS >= 500$.
  - **Maximum number of particles to diffuse into the lattice (N)** :
    In our experiments, we add $N >= 50000$ particles into the square lattice or until the diffusion locus is exhausted.
  - **Stickiness(k)** :
    In our experiments, we limit to $k = [0.05, 0.025, 0.0125, 0.00625, 0.003125, 0.0015625]$ i.e $k = [1e^{-3}, 5e^{-2}]]$. Also experiments were run with $k = [1.0, 0.5, 0.1]$ to observe the behavior of the Brownian tree generation.

# Simulation configurations

- Configurations of simulation are:
  - **Lattice Structure** : Shape of the lattice.
    In our experiments, we limit to only the square lattice.
  - **Diffusion Locus** : Locus of points used to diffuse the particles into the lattice.
    In our experiments, we consider randomly generating a new particle along the perimeter of the square lattice.
  - **Seed particle location** : Starting location of the seed particle in the lattice.
    In our experiments, we place the seed particle at the center of an empty square lattice.
  - **Boundary condition on the particles** : The behavior of the particles on reaching the boundary.
    In our experiments, we reintroduce the particle into the lattice in a toroidally bound manner until it's stuck.

# Overcoming simulation bottleneck

- The whole Experimental setup was modeled in python and can be found in github : DLA - Diffusion Limited Aggregation.
- Even though the experimental procedure for developing Brownian trees are simple, it was observed that the simulation time was the main bottleneck for data logging.
- The time complexity for the simulation is of the order of $O(LS^2 * N * k^{-a})$ where a=[1, 2] .
- For $LS >= 500, N >= 50000$ and $k = [1e^{-3}, 5e{-}2]$, developing Brownian tree for each combinations was exponentially high.
- The run-time even increases further as the lattice size increase due to the increase in random walk steps. Lower stickiness (k) further increases the amount of random walks steps due to decreased stickiness.

# Solution : Diffusion locus mapping

- Observing closely the plots of the number of random steps, one main factor for in reducing the time can be by reducing the diffusion locus.

- It can be seen that any particle which is randomly generated along LC1 is equally likely to end up on LC2 during the random walk due to the torroidally bound lattice. ($LC2 << LC1$, 2)
- This way we can reduce the walking space for the particle without compromising in the generation of Brownian tree.

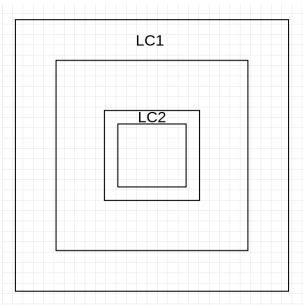
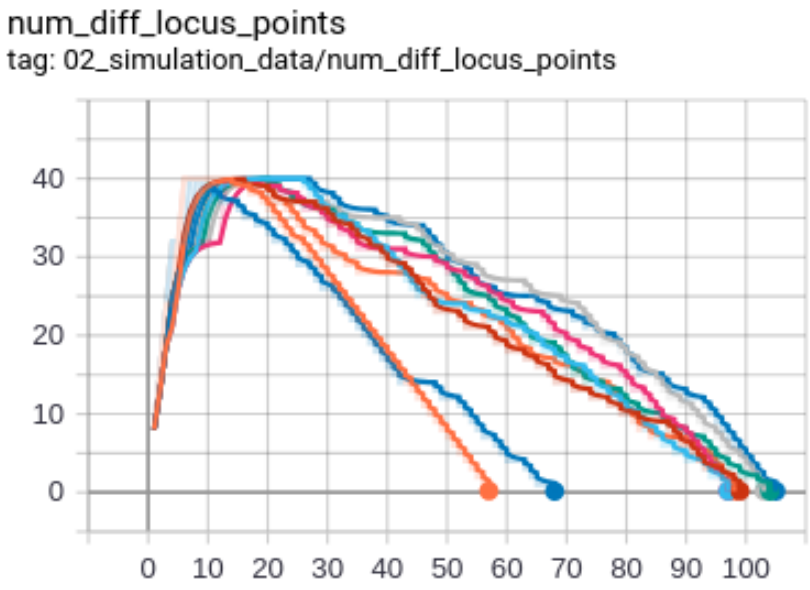



Figure: LC1 indicates the actual diffusion locus along the perimeter of the lattice. LC2 indicates the perimeter of smallest square enclosing all the particles in the lattice.

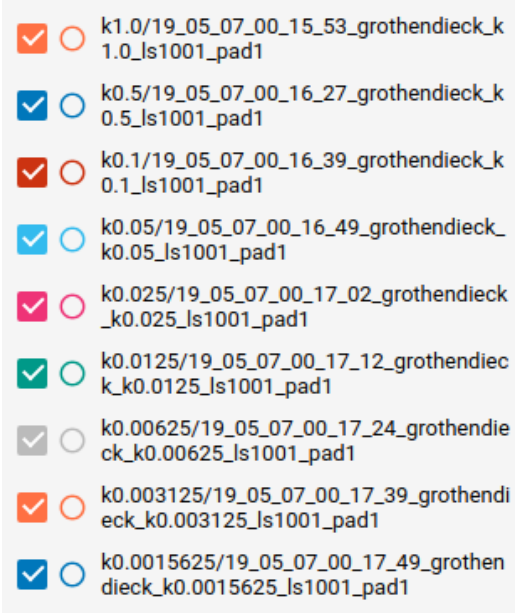
# Solution : Diffusion locus mapping

- This simple inference helped in generating the trees with progressively increasing the lattice size by 2 units and adding the particles in the minimized diffusion locus LC2.
- This procedure can be visualized as unwinding the main torus into a square lattice, extracting the smallest square lattice with all the particles, pad 1 units of extra lattice on all sides of this smaller lattice and rewind it back into a torus for diffusion and aggregation.
- We perform this procedure each time the base lattice grows to accommodate the progressive growth of lattice.

# Solution : Diffusion locus mapping (continued ...)




(a) Number of diffusion locus locations vs N




(b) legend

Figure: Number of diffusion locus points v/s Number of particles in the lattice. The number of diffusion locus points increasing progressively as the tree keeps growing thereby reducing the amount of random walk required. The number of diffusion locus points later decreases due to the particles getting stuck on the diffusion locus causing termination.

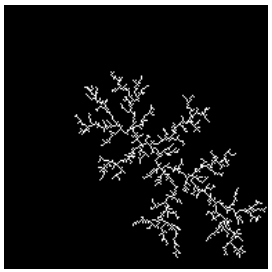
# Solution : Diffusion locus mapping (runtimes)

| **stickiness** | **without locus mapping** | **with locus mapping** | **speed up** |
|---|---|---|---|
| k=0.05 | 6.05 | **0.11** | **55x** |
| k=0.025 | 6.65 | **0.15** | **44x** |
| k=0.0125 | 7.43 | **0.18** | **41x** |
| k=0.00625 | 8.83 | **0.21** | **42x** |
| k=0.003125 | 12.03 | **0.51** | **23x** |
| k=0.0015625 | 22.20 | **0.80** | **27x** |

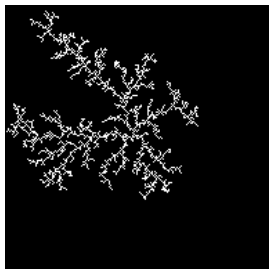
Table: Brownian tree generation time (in hours) for $N = 1000$ particles in the lattice.

It is important to observe that at-least a significant **20x** speed up is observed in all the cases which is a very huge improvement.

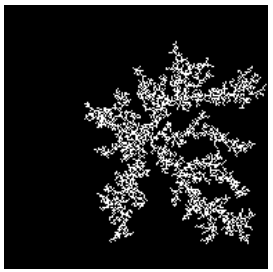
# Brownian tree versus stickiness (k)



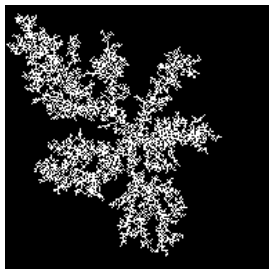
(a) k=1.0



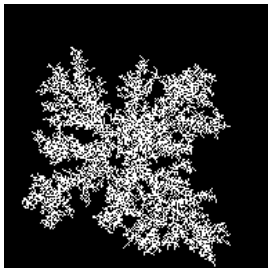
(b) k=0.5



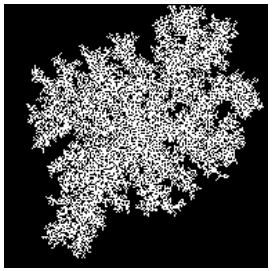
(c) k=0.1



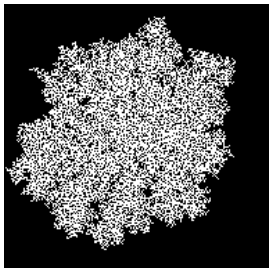
(d) k=0.05



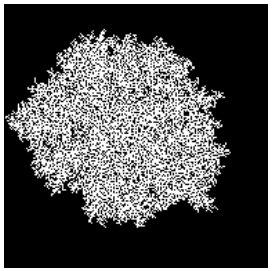
(e) k=0.025



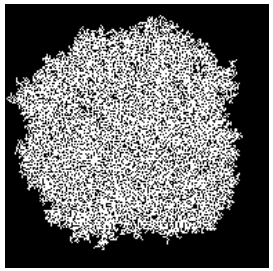
(f) k=0.0125



(g) k=0.00625



(h) k=0.003125



(i) k=0.0015625

Figure: Different Brownian trees formed with different stickiness (k) for the same lattice size of 200. Lower stickiness results in more densely clustered trees.

# Stickiness (k) = 1.0

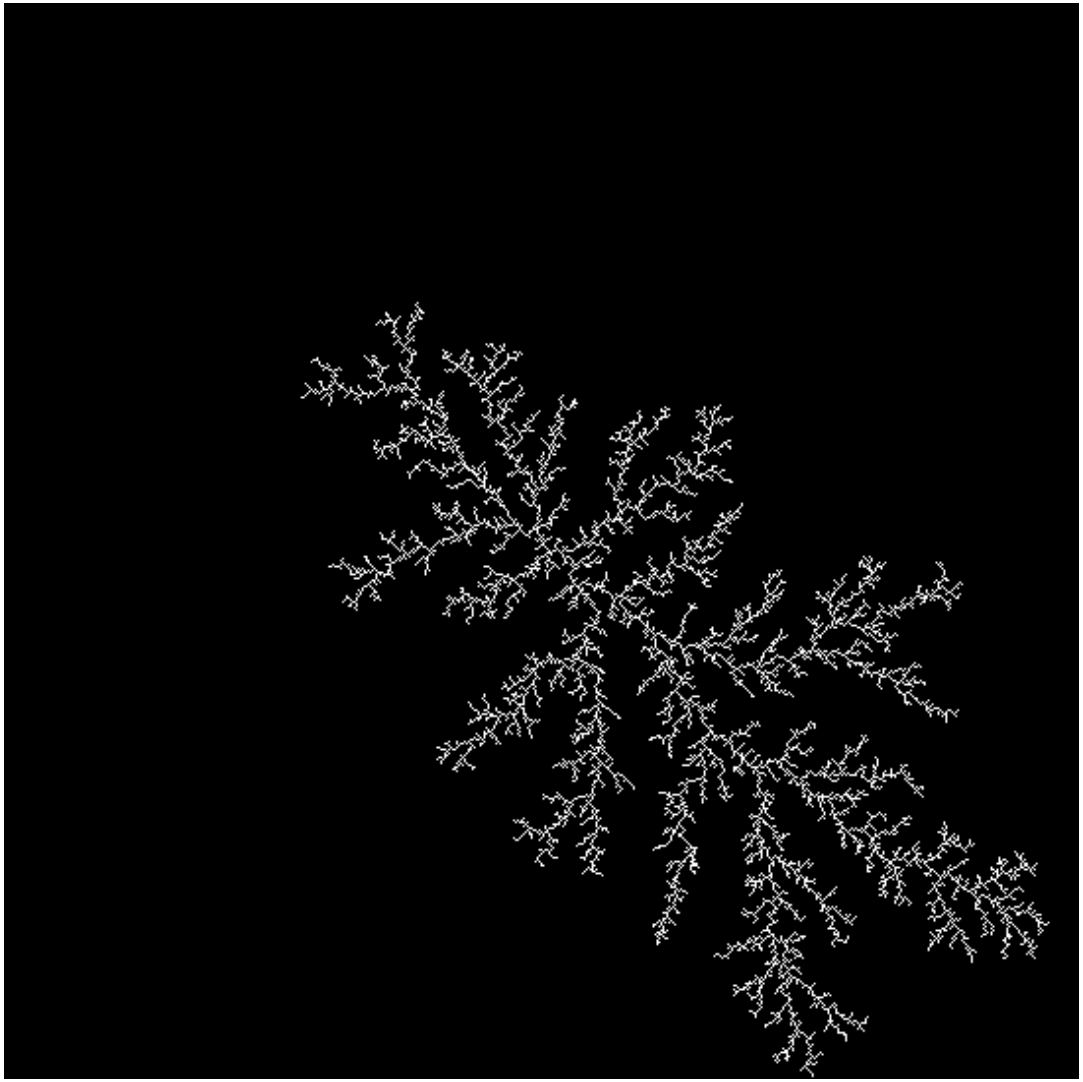


Figure: Stickiness (k) = 1.0

# Stickiness (k) = 0.5

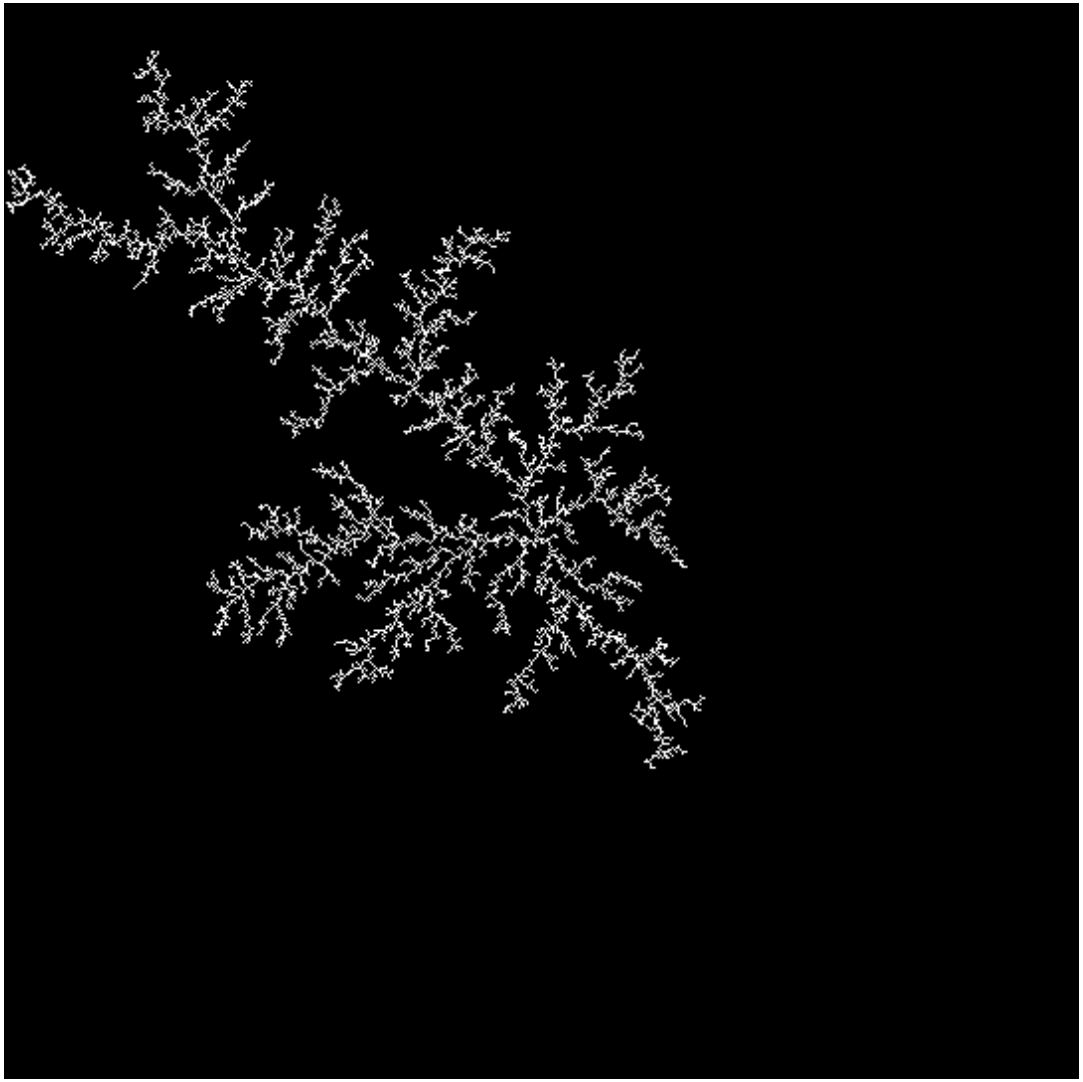


Figure: Stickiness (k) = 0.5

# Stickiness (k) = 0.1

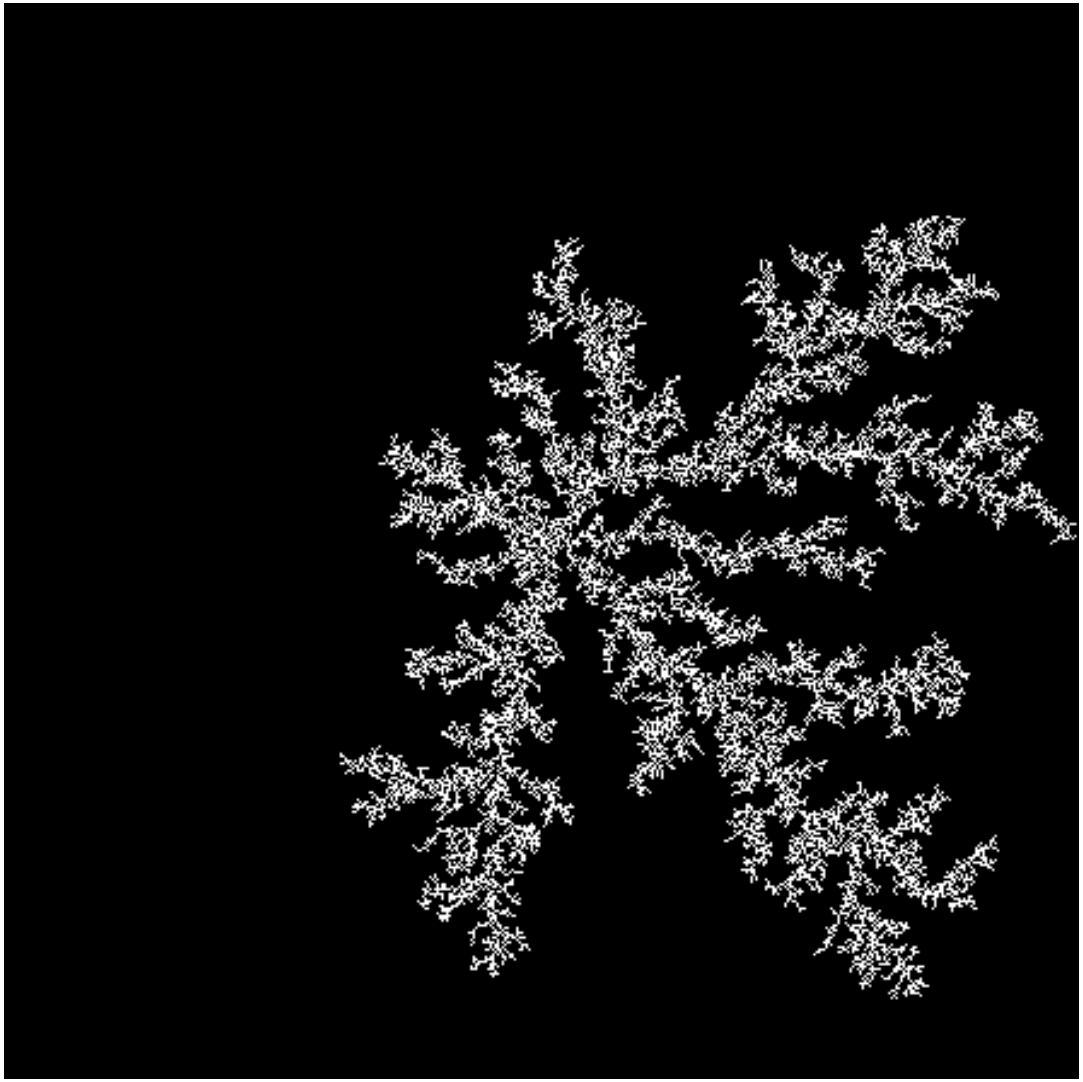


Figure: Stickiness (k) = 0.1

# Stickiness (k) = 0.05

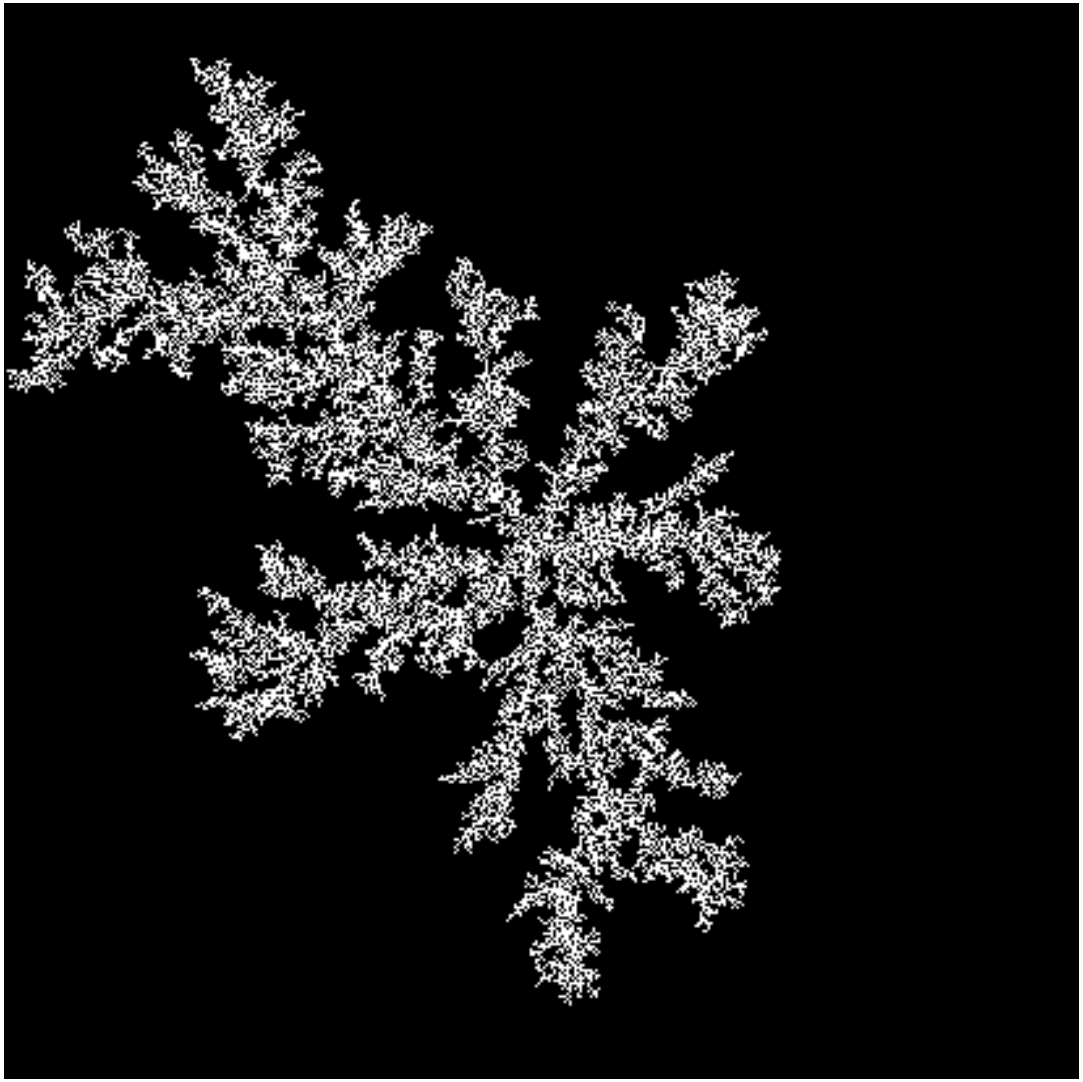


Figure: Stickiness (k) = 0.05

# Stickiness (k) = 0.025

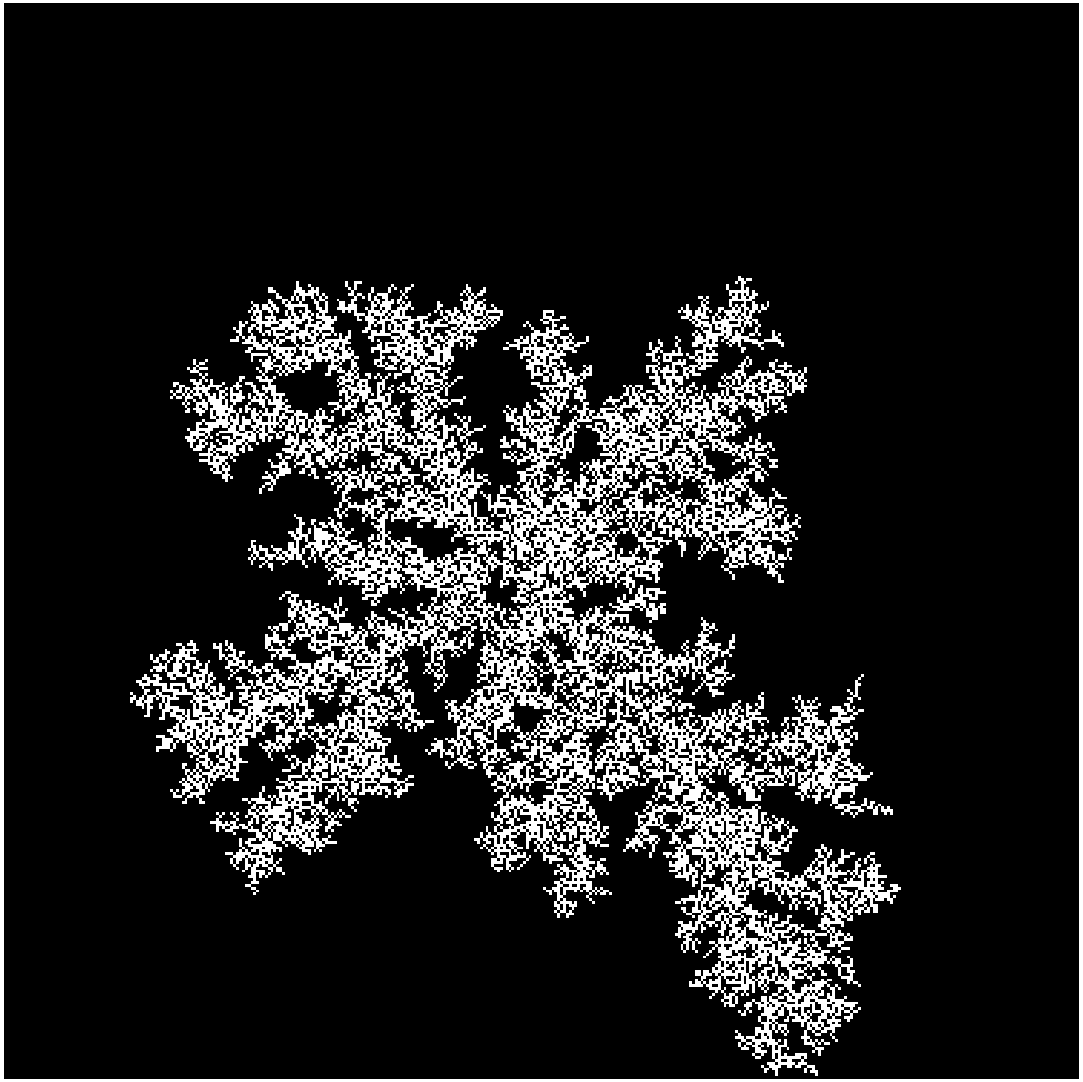


Figure: Stickiness (k) = 0.025

# Stickiness (k) = 0.0125

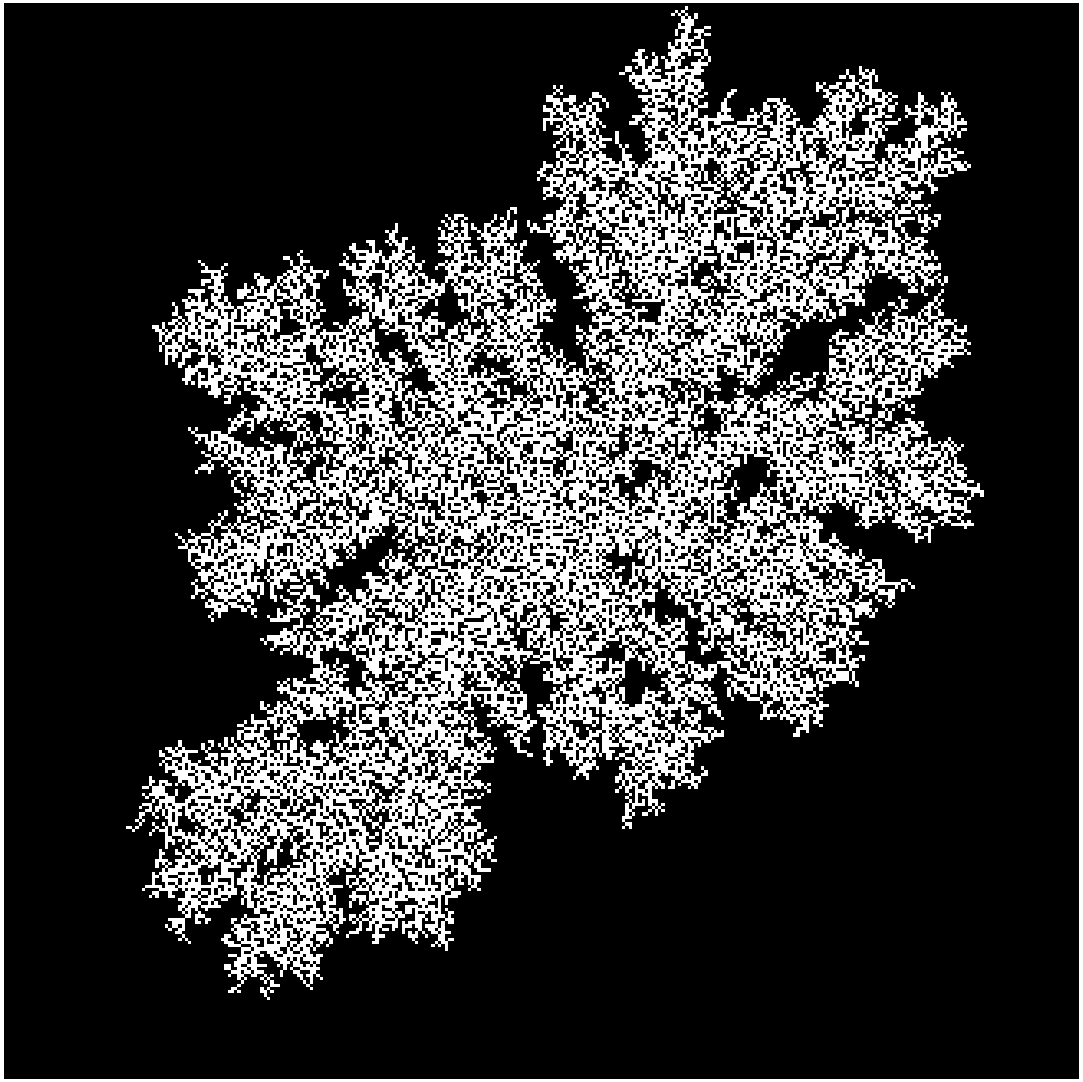


Figure: Stickiness (k) = 0.0125

# Stickiness (k) = 0.00625

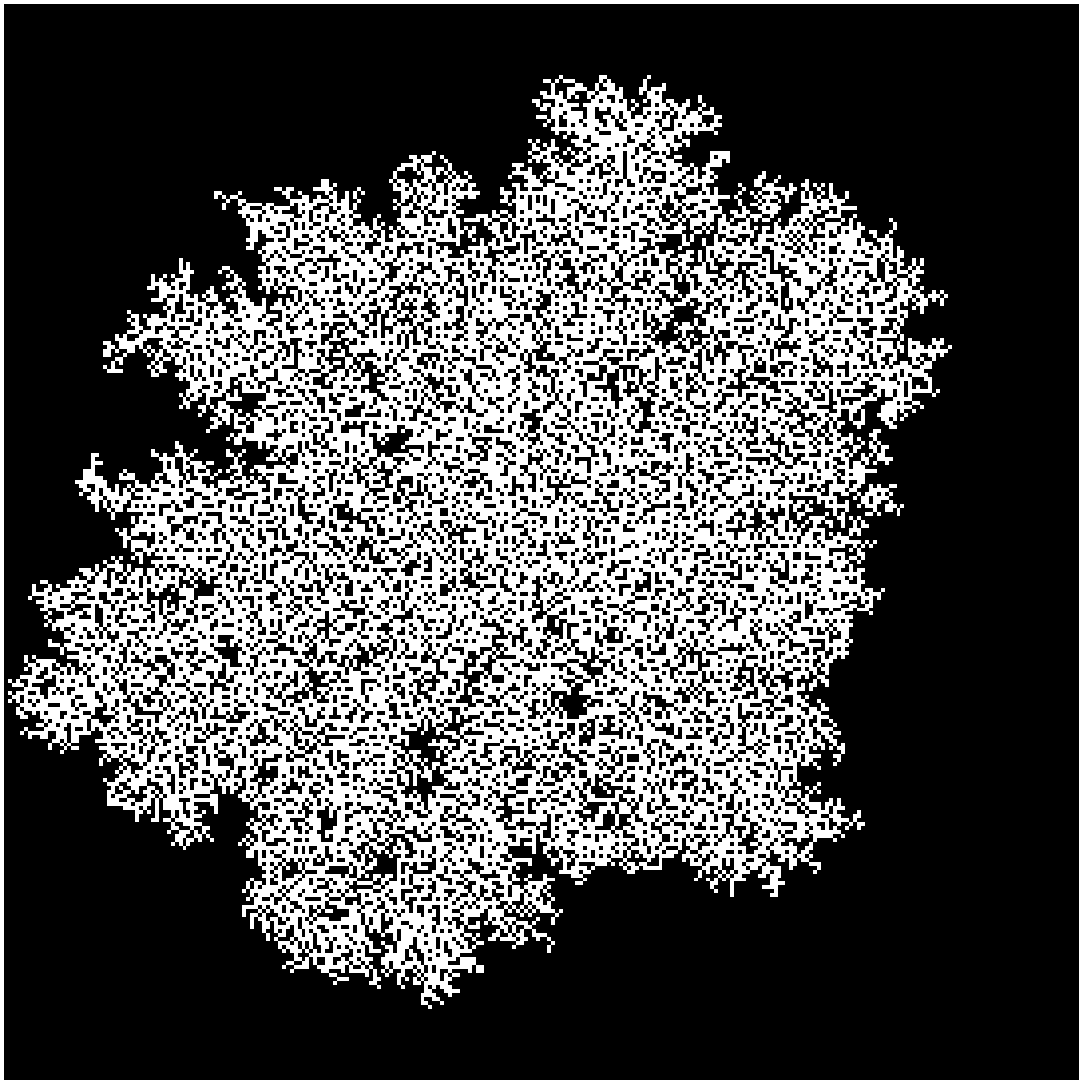


Figure: Stickiness (k) = 0.00625

# Stickiness (k) = 0.003125

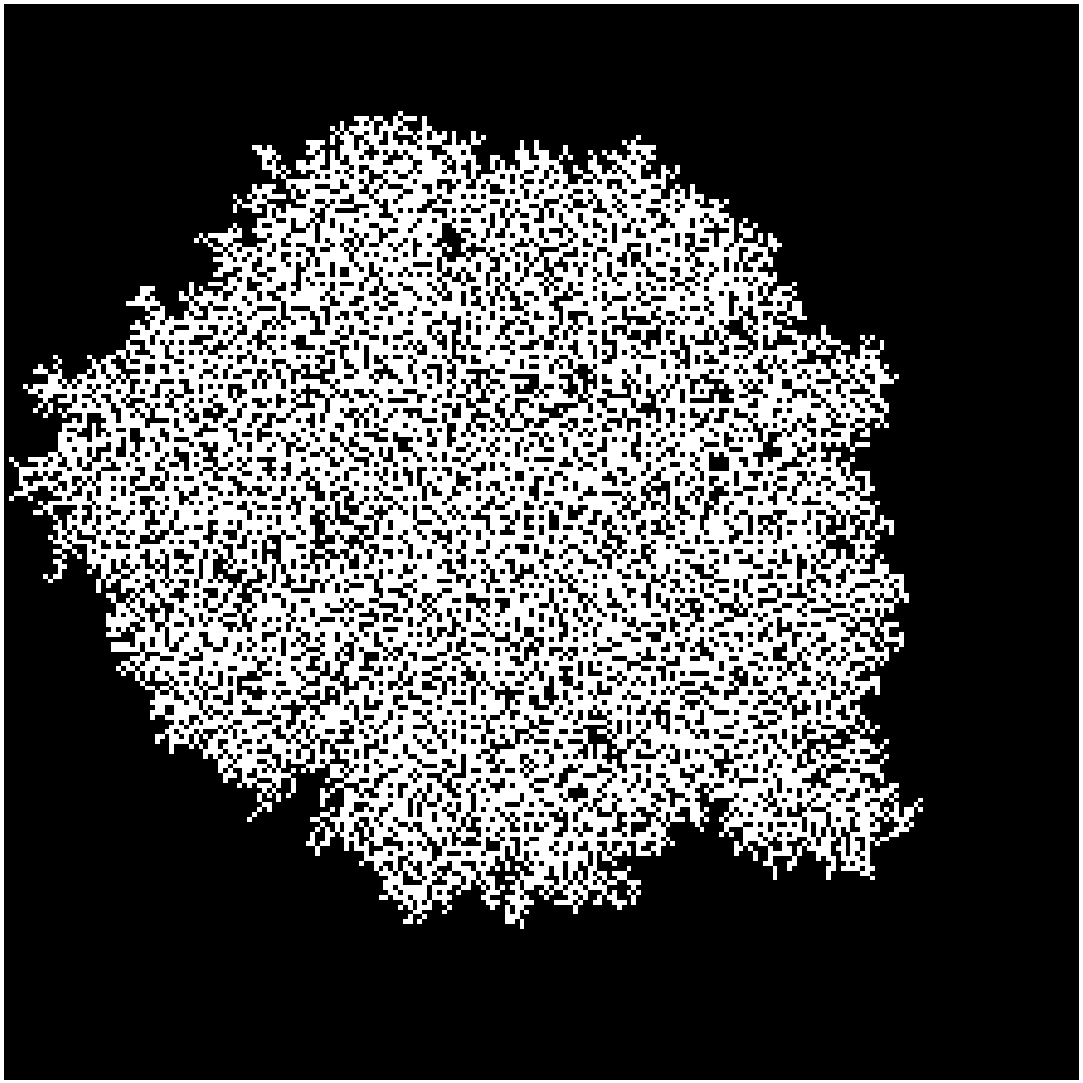


Figure: Stickiness (k) = 0.003125

# Stickiness (k) = 0.0015625

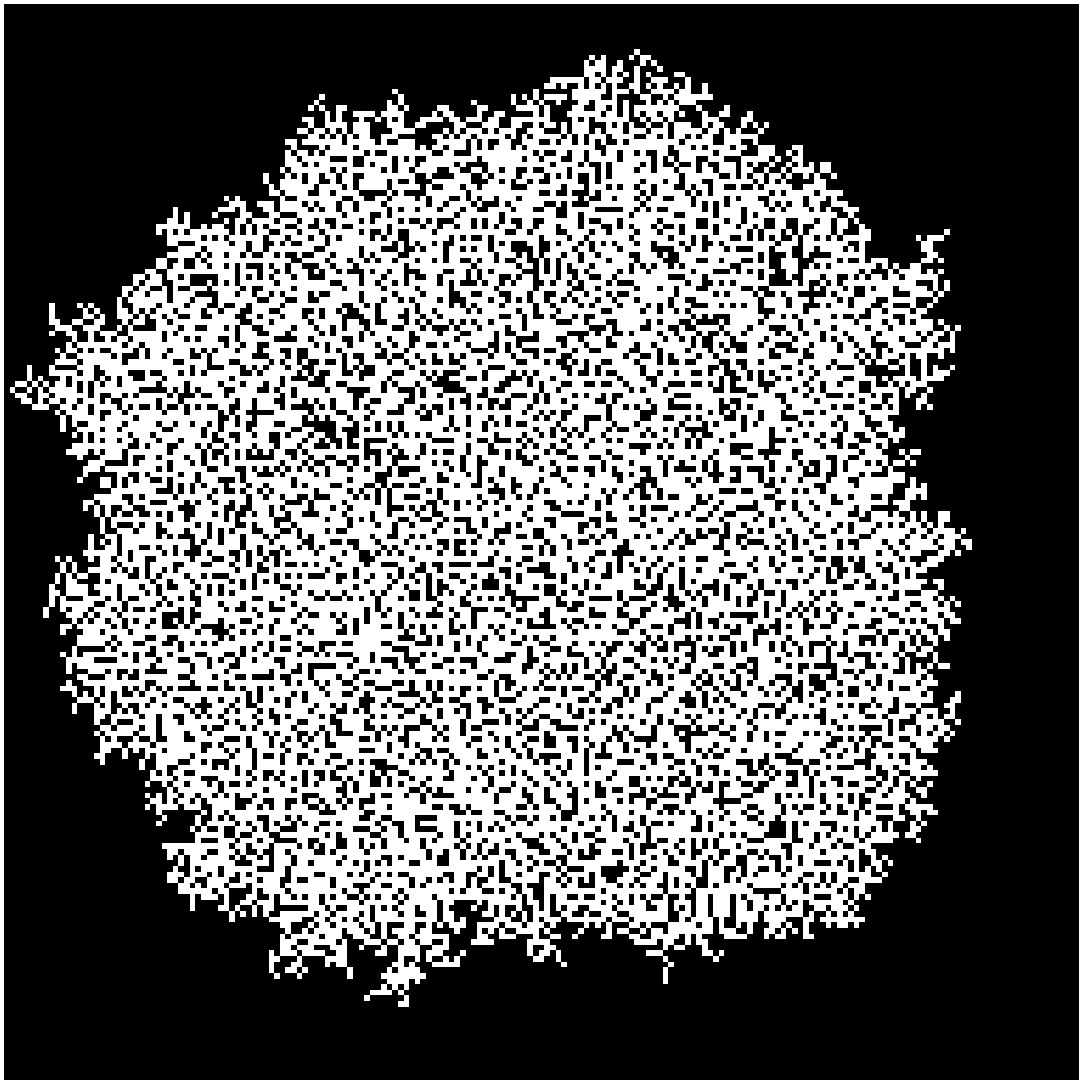


Figure: Stickiness (k) = 0.0015625

# Feature for estimation of stickiness (k)

- The data from the previous Brownian tree generation part are logged during the simulation.
- From a given image with a Brownian tree, we only obtain the coordinates of the particles and the number of particles.
- To estimate k using any machine learning even as simple as a linear regression we need more number of features (independent variable).
- From the Experimental observations, we try to obtain various information (described in 2) apart from the number of particles.
- We run the simulations for $LS = 1001$, $k = [0.05, .025, 0.0125, 0.00625, 0.003125, 0.0015625]$, $N = 1001^2$ twice and collect the required features for around 2 million datapoints to estimate the value of k for a given image.

# Feature for estimation of stickiness (k) (continued ...)

- From Figures(4,5,6,7,8,9,10,11,12,13) , it can be clearly seen that the cluster density is a function of stickiness (k).
- Also, the effective outer perimeter available for aggregation is also a function of stickiness (k).
- Observing these we extract possible features from the simulation to create a dataset for estimation of stickiness (k).
- The **external contour** of the tree is the most informative entity in identifying the stickiness as it encapsulates almost all the information of the tree.
- Next, we discuss on the **contour features** helpful in predicting $k$. We extract these features from all the trees and log them to create a dataset.

# Feature for estimation of stickiness (k) (continued ...

## Contour perimeter

The effective perimeter of the tree available for aggregation.

## Contour area

The effective area enclosed by the perimeter of the tree.

## Rectangular bounding box area

The area of the minimum rectangular bounding box enclosing the tree.

## Radius of bounding circle

The radius of the minimum circle bounding enclosing the tree.

# Feature for estimation of stickiness (k) (continued ...)

## Convex hull area

The area of the convex hull enclosing the tree.

## Extent

Extent is defined as the ratio of contour area to bounding rectangle area.

## Solidity

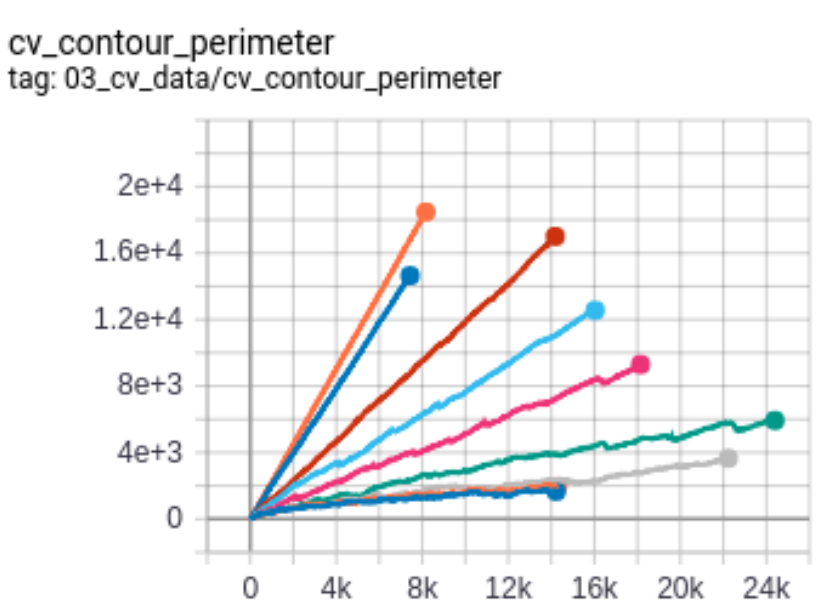
Solidity is defined as the ratio of contour area to its convex hull area.

## Fractal Dimension

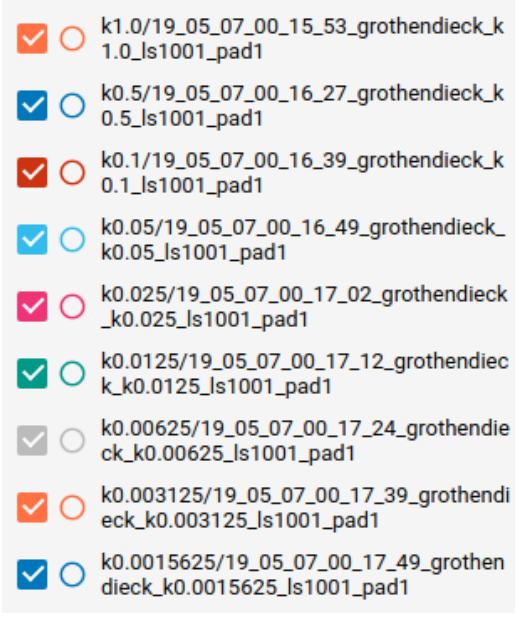
Fractal Dimension ($D_f$) is estimated using the standard box counting method as $D_f$ is a function of stickiness (k) as well.

# Feature analysis : Contour perimeter

The effective perimeter of the tree available for aggregation.




(a) Contour perimeter vs N

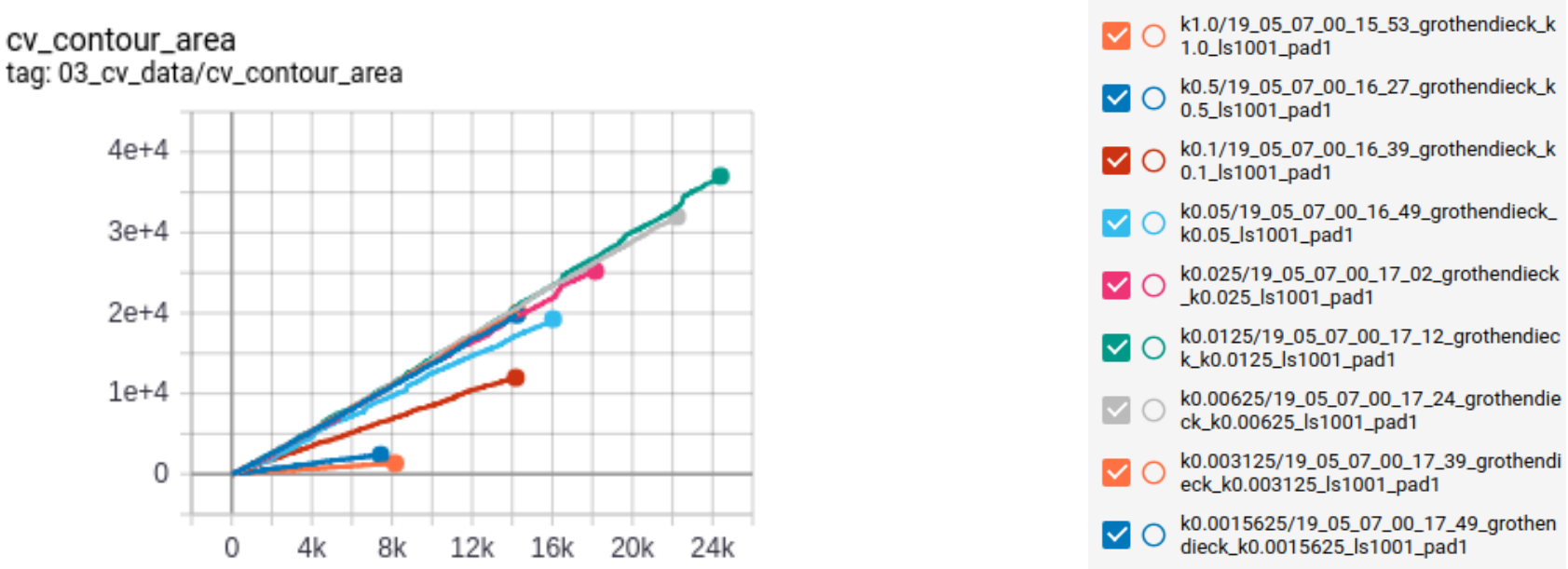



(b) legend

Figure: Contour perimeter v/s Number of particles in the lattice.

# Feature analysis : Contour area

The effective area enclosed by the perimeter of the tree.



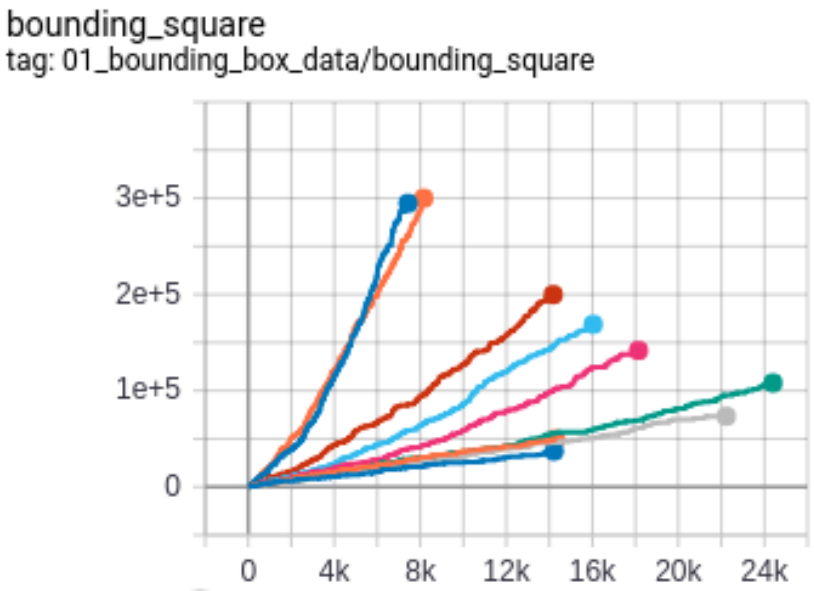

(a) Contour area vs N

(b) legend

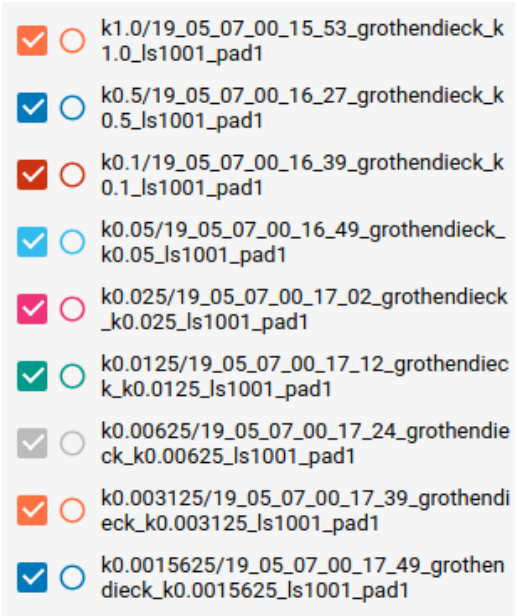
Figure: Contour area v/s Number of particles in the lattice.

# Feature analysis : Rectangular bounding box area

The area of the minimum rectangular bounding box enclosing the tree.




(a) Rectangular bounding box area vs N

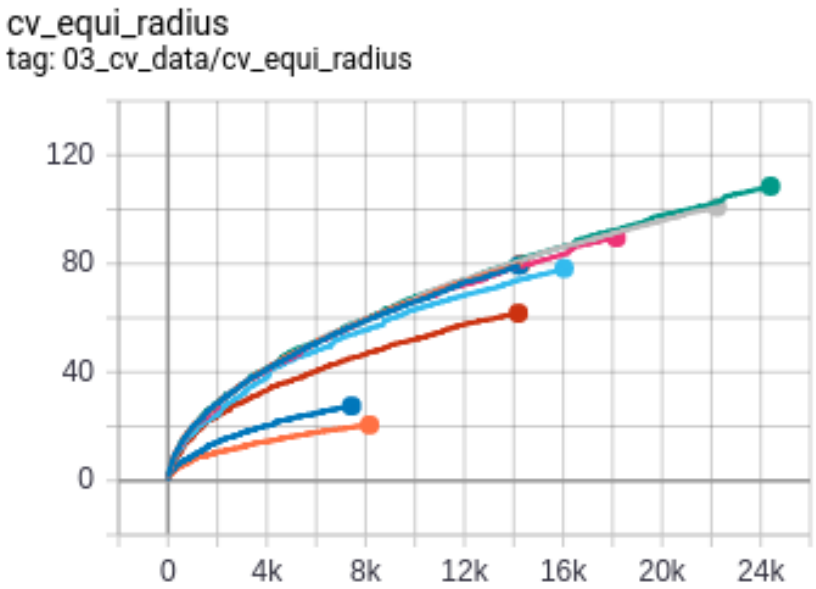



(b) legend

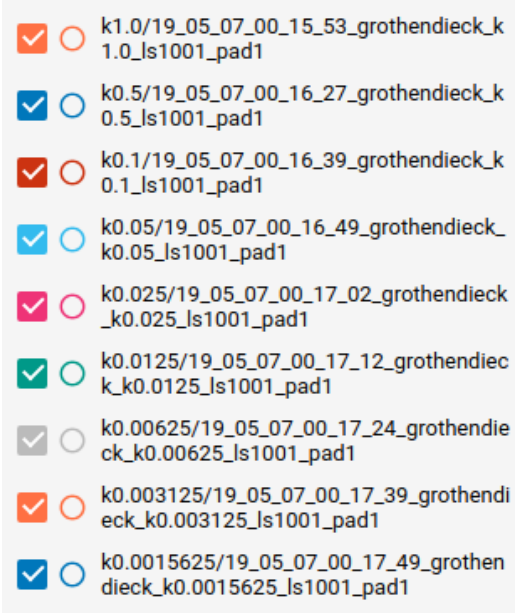
Figure: Rectangular bounding box area v/s Number of particles in the lattice.

# Feature analysis : Radius of bounding circle

The radius of the minimum circle bounding enclosing the tree.




(a) Radius of bounding circle vs N

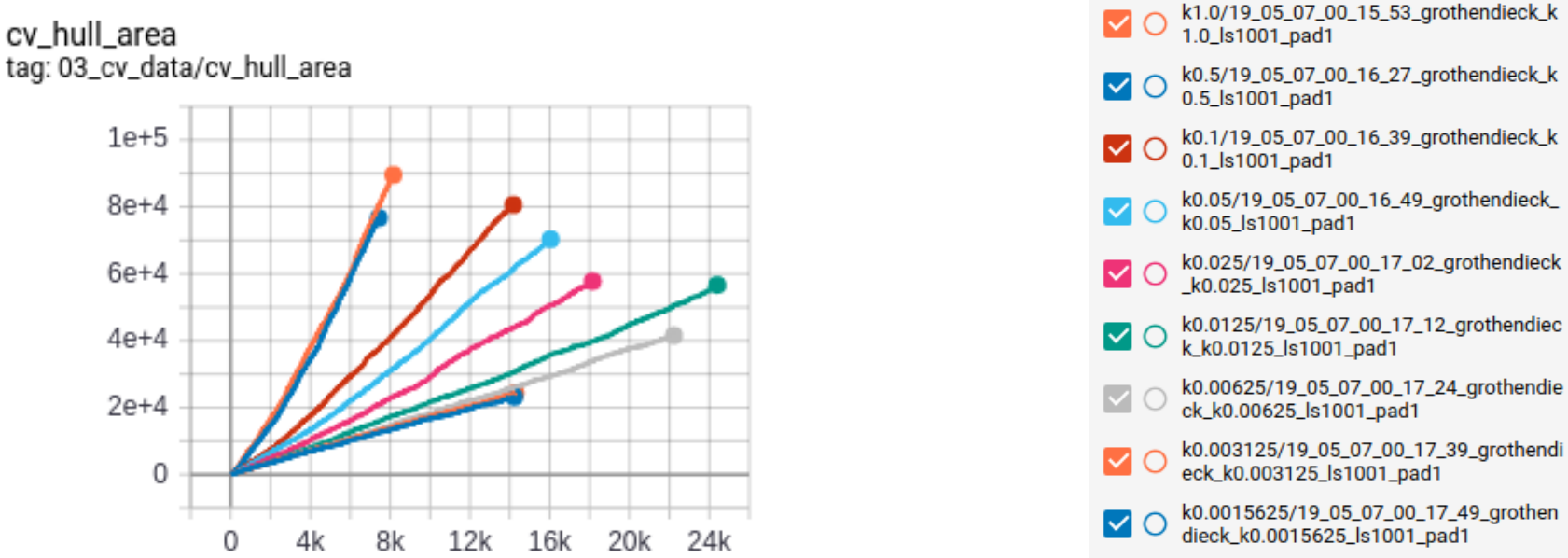



(b) legend

Figure: Radius of bounding circle v/s Number of particles in the lattice.

# Feature analysis : Convex hull area

The area of the convex hull enclosing the tree.

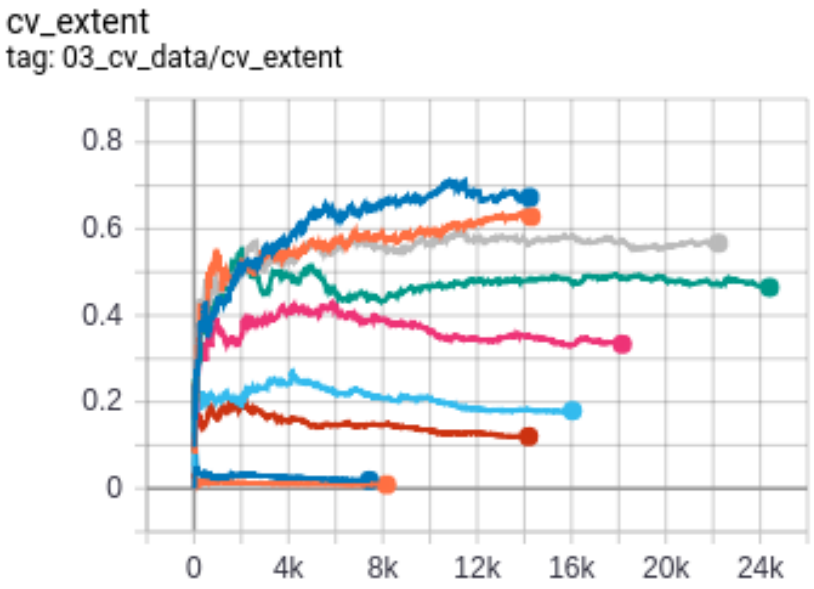



(a) Convex hull area vs N (b) legend

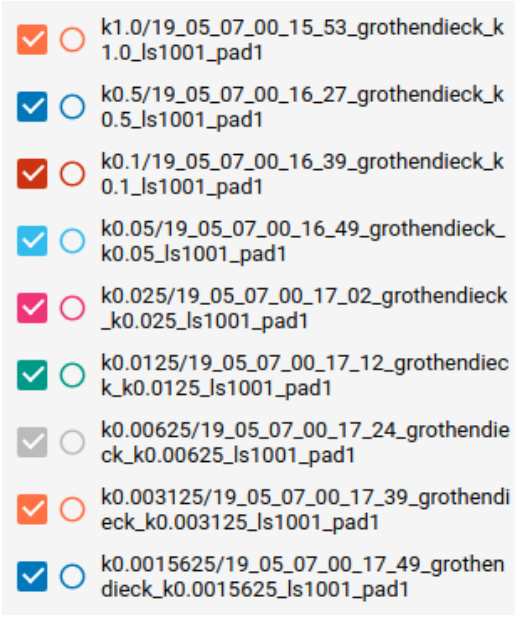
Figure: Convex hull area v/s Number of particles in the lattice.

# Feature analysis : Extent

Extent is defined as the ratio of contour area to bounding rectangle area.




(a) Extent vs N

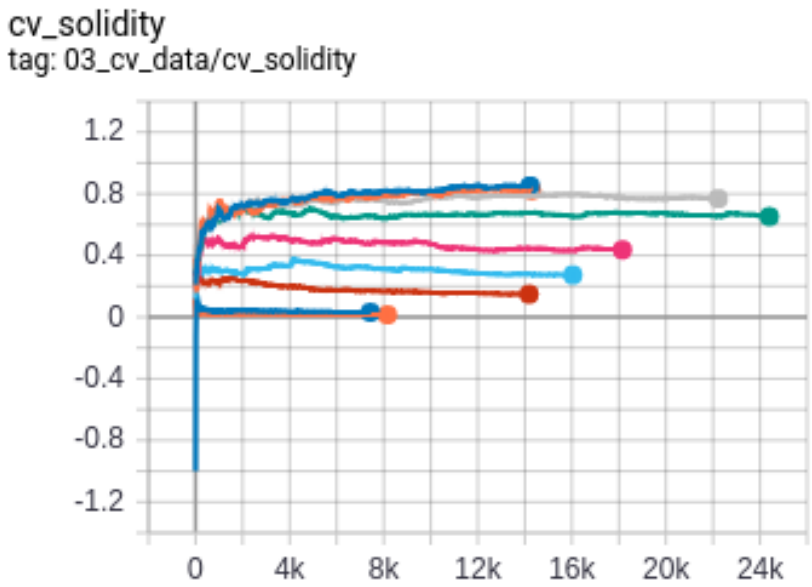



(b) legend

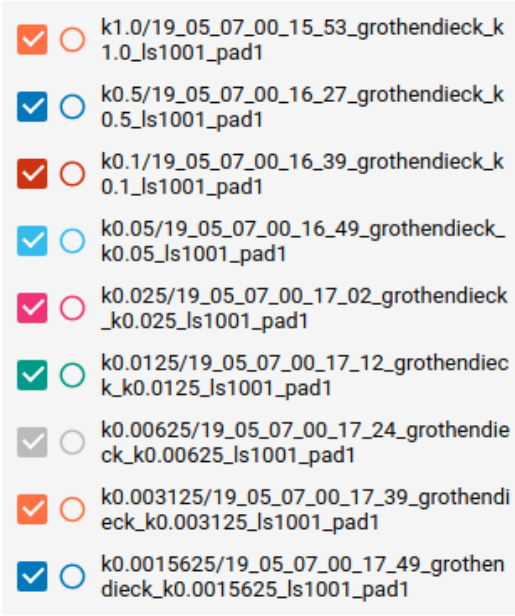
Figure: Extent v/s Number of particles in the lattice.

# Feature analysis : Solidity

Solidity is defined as the ratio of contour area to its convex hull area.




(a) Solidity vs N

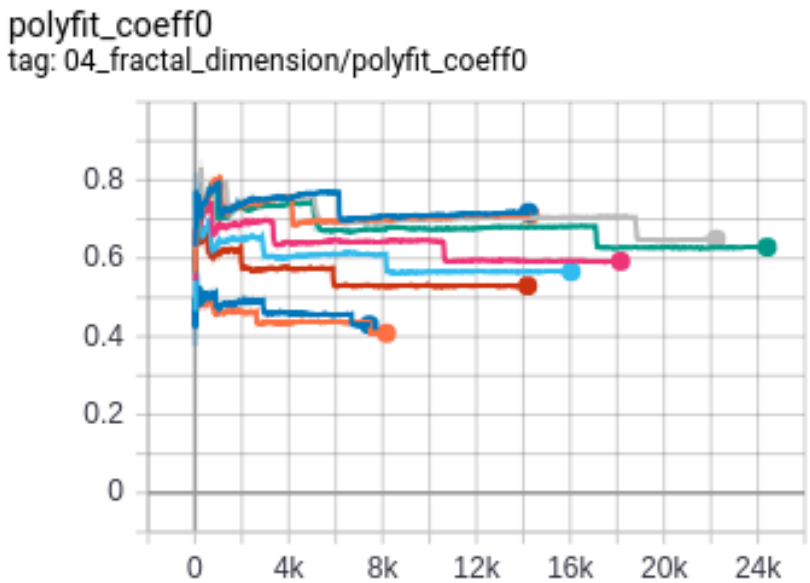



(b) legend

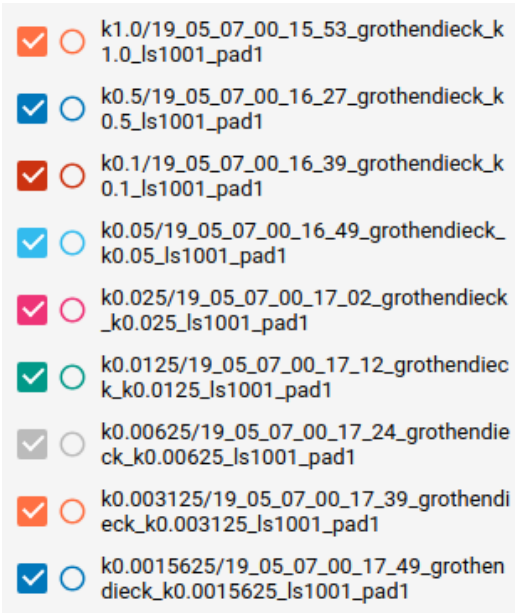
Figure: Solidity v/s Number of particles in the lattice.

# Feature analysis : Fractal Dimension

Fractal Dimension ($D_f$) is estimated using the standard box counting method as $D_f$ is a function of stickiness (k) as well.




(a) Fractal Dimension vs N




(b) legend

Figure: Fractal Dimension ($D_f$) v/s Number of particles in the lattice.

# Feature analysis : Correlation plot

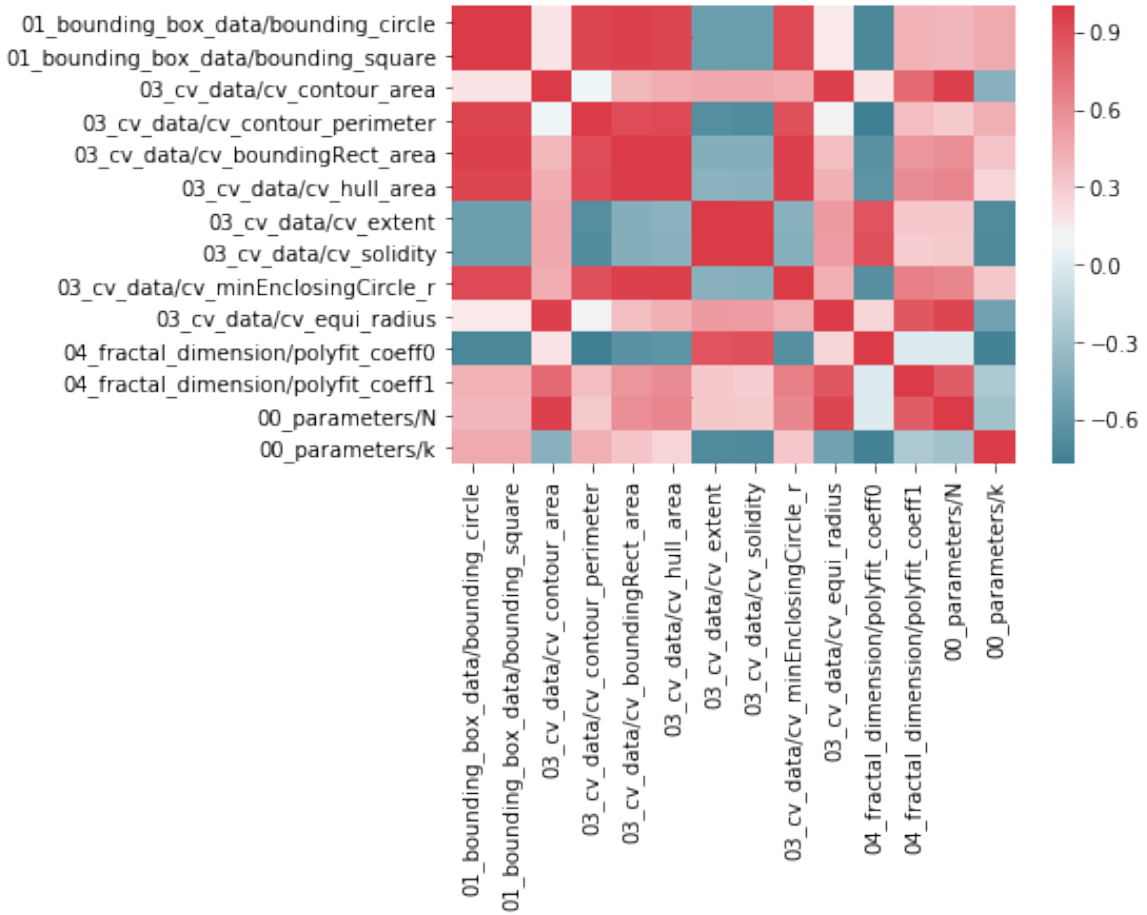
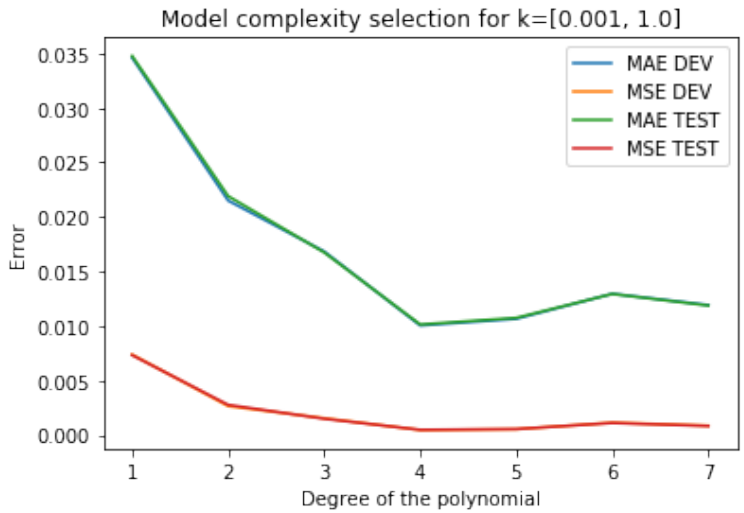



Figure: Analysis of correlation of features. Features such as $D_f$, solidity, extent, perimeter are in high correlation with $k$.

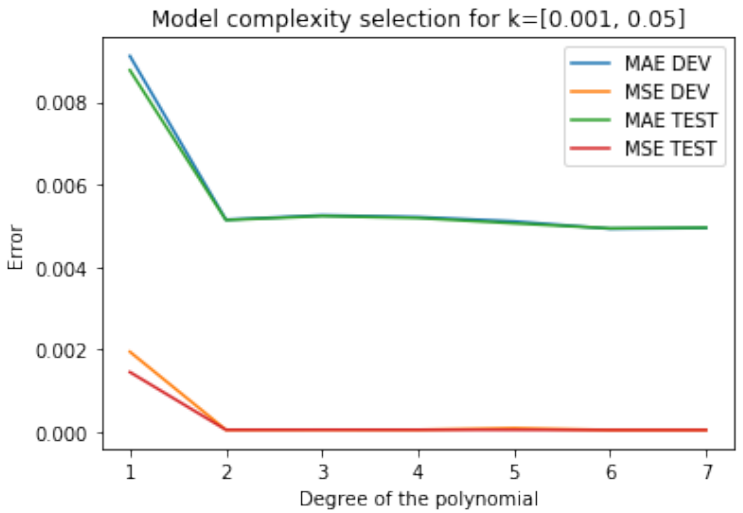
# Model selection

- Observing the features from before, The relation of stickiness (k) with $N$ is clearly nonlinear.
- Fig.14 indicates the slope of contour perimeter v/s $N$ is a clear function of $k$ (non linear as the plots are not equally spaced with $k$).
- We randomly shuffle and split dataset into train (66%) and test (33%). Further train is split into train (80%) and dev (20%) for identifying the best model complexity with least dev set error .
- We fit a polynomial curve $\frac{contour\_perimeter}{N}$ against $log(k)$ and observe a very good prediction of $k$ on the test-set.
- We use $MAE$ (mean absolute error) and $MSE$ (mean squared error) as evaluation metric for Model selection.

# Model selection (continued ...)




(a)




(b)

Figure: Model complexity for different ranges of $k$.
Best model for (a) $k = [1e^{-3}, 1.0]$ and (b) $k = [1e^{-3}, 5e^{-2}]$ are degree 4 and 2 respectively.

# Results and Conclusion

$$MAE_{range} = \frac{MAE}{k_{max} - k_{min}}, MSE_{range} = \frac{MSE}{k_{max} - k_{min}} \quad (7)$$

| | $k = [1e^{-3}, 1.0]$ | | $k = [1e^{-3}, 5e^{-2}]$ | |
|---|---|---|---|---|
| | DEV | TEST | DEV | TEST |
| $MAE$ | $1004.17e^{-5}$ | $1013.55e^{-5}$ | $494.50e^{-5}$ | $495.40e^{-5}$ |
| $MSE$ | $45.37e^{-5}$ | $47.37e^{-5}$ | $4.98e^{-5}$ | $4.97e^{-5}$ |
| $MAE_{range}$ | $1005.18e^{-5}$ | $1014.57e^{-5}$ | $10091.86e^{-5}$ | $10110.27e^{-5}$ |
| $MSE_{range}$ | $45.42e^{-5}$ | $47.42e^{-5}$ | $101.81e^{-5}$ | $101.61e^{-5}$ |

Table: $MAE$ and $MSE$ decreases with decrease in the range as the model best fits for that particular range. $MAE_{range}$ and $MSE_{range}$ increases with decrease in the range because for smaller range this measure expects higher precision.

# Future work

- **Is this Enough?! Are we done?!**
- We can try to get further insights from the other logged data features.
- Using the insights, we can included these features and fit more complex models such as SVR (Support Vector Regressors with RBF kernel) or linear regression models with L1 and L2 regularizers.

# The End