PSet1 Report

Ali Abolhassanzadeh Mahani

October 17, 2020

1 The Koch Fractal Set

I used the `Turtle` tool in the `python` library set to draw the *koch fractal set*. I made a class called `Koch` that takes in the `level` and `max_level` as inputs and creates the koch fractal recursively. Then, there's the `draw()` method in `Koch` that gets called recursively to draw the koch fractal. I drew it to level 5 and output used a snipping tool to take a sceenshot of it.(Fig1)

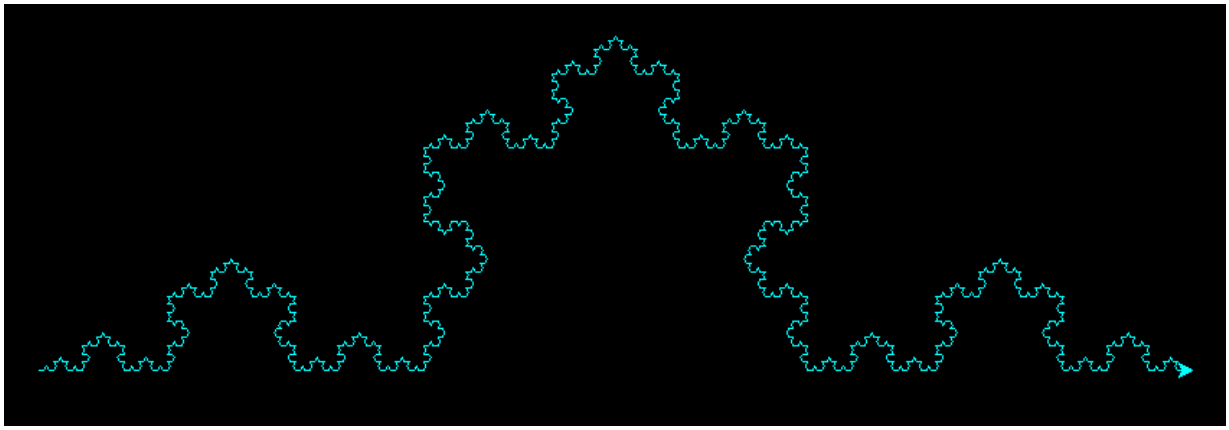

Figure 1: Koch fractal set, drawn to level 5. The length is reaching the pixel size now.

2 The Serpinski Fractal Set

2.1 Deterministic

I created a class for `Serpinski` and a module to `add_subset`. Then a module to recursively go through the `sub_sets` and draws them using `plt.fill()`. This method is very inefficient. A better way is to append the coordinates to a single array recursively and then draw it. This way makes it much faster. plt is so damn slow! (:face_palming).

The generated set with 8 levels of subsets is shown in Fig2.1.

2.2 Randomized

Here I added a module to the `Serpinski` class called `is_bound()` which takes the coordinates of the point and recursively justifies if the point is in the serpinski fractal boundary. If yes, then I save the point coordinates. After I successfully find 50000 dots, I draw them using `plt.scatter(x, y, s=0.1)` where `s=0.1` refers to the markersize which is set to 0.1 pt. Now Here's the Simulation for the level 8 fractal from the previous subsection. (Fig 2.2)

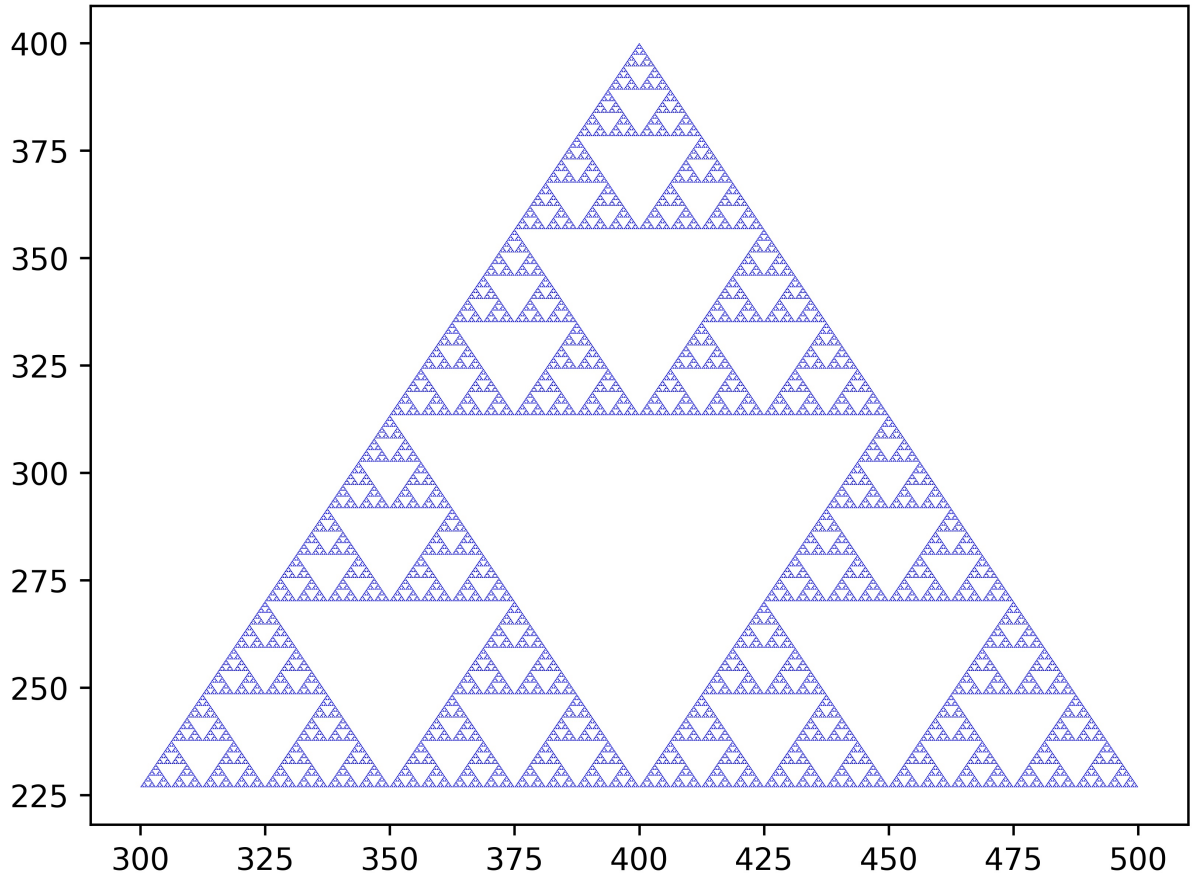

Figure 2: The level 8 serpinski fractal set.

3 Random Deposition

3.1 Intro

In oder to work with Random Depostion, I made a function that generates the canvas, `n` times. This fucntion calls another function that generates the canvas *once*. This, then, calls a function that deploys a particle using the rule that it's been given – e.g. the Ballistic Deposition rule or the Competitve BD rule.

Equations to be used:

!! Note: L is the length of the surface, t is time, w is width (roughness)

$$w(L,t) = t^{\beta},\ t \ll t_s \tag{1}$$

$$t_s = L^z \tag{2}$$

$$w_s = t_s^{\beta} = L^{\beta z} = L^{\alpha} \tag{3}$$

After generating the graphics once, I shutdown (`comment out`) the graphics part and do the calculations. I fit the curve using polyfit and reported the value for β

3.2 Random Deposition

The graphics to lay 25 layers is shown in Fig 3.2 The value for β using eq.1 is as follows:

$$\beta = 0.44 \pm 0.02$$

And the plot is shown at Fig.3.2

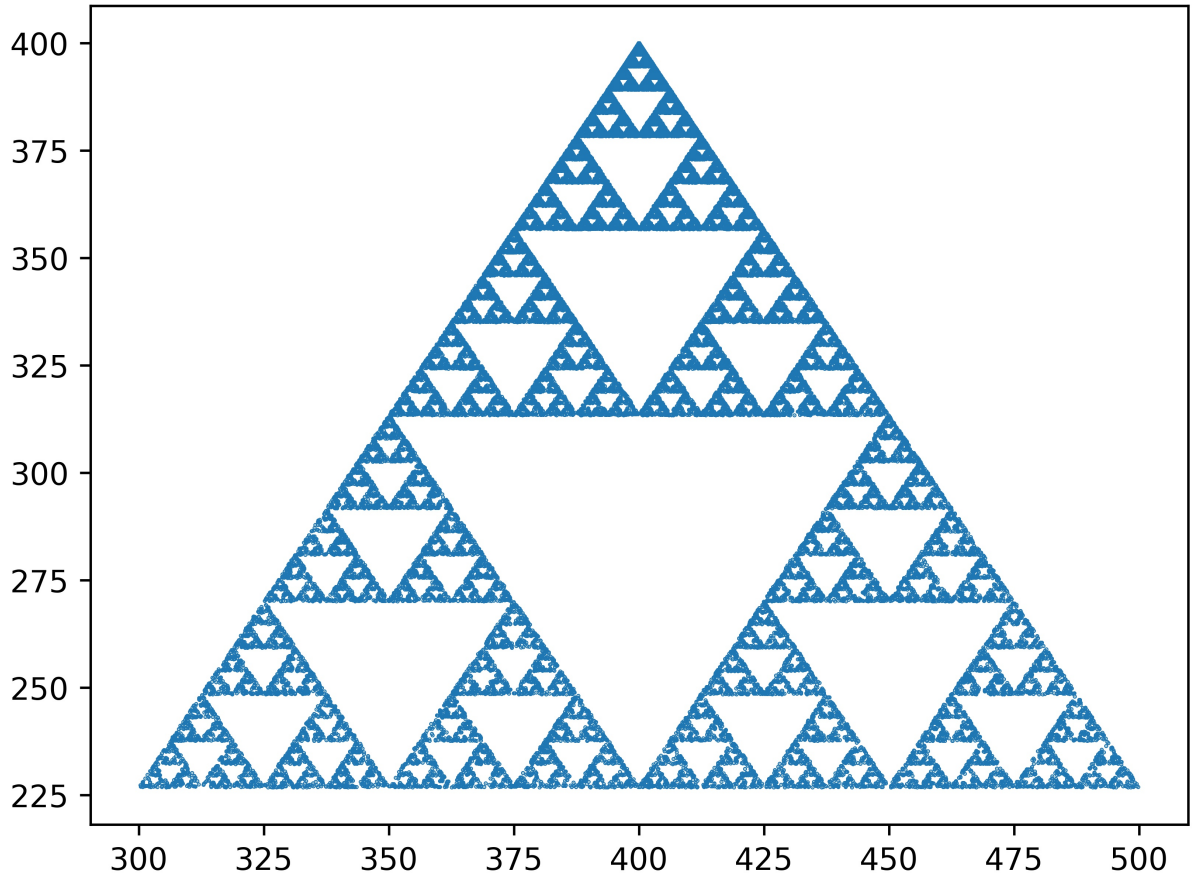

Figure 3: find 50000 radom particles for the level 8 serpinski fractal from Fig2.1

3.3 Random Deposition with Surface Relaxation

The canvas for this part is shown in Fig3.3. In order to reach saturation we need at least about 50000 particles. to make the plot (Fig3.3) I ran the simulation 10 times with 800 layer deposits total. The value for t_s was output by looking at the graph and getting the data that was closest to the values we were looking for. The value found for beta is:

$$\beta = 1.64 \pm 0.06$$

The values for length L, saturation width w_s and saturation time t_s is as follows:

$$\begin{aligned} L &= [10, 50, 100, 150, 200, 250] \\ t_s &= [32, 3072, 16384, 49152, 131072, 1048276] \\ w_s &= [0.6, 1.1, 1.678, 2.725, 2.789, 3.558] \end{aligned}$$

These values are plotted using the file `plots.py`. Plot fittings also happen there.
Values found for *beta* and z are as follows:

$$\beta = 0.18 \pm 0.02, \qquad z = 2.97 \pm 0.24, \qquad \Rightarrow \alpha \simeq 0.547$$

plots for *beta* and z are found in Fig3.3

The theoretical value for β is 0.24 and alpha, 0.47 which gives a percentile error of $\eta_\beta = 25\%$ and $\eta_\alpha = \%$ respectively

3.4 Ballistic Deposition

The canvas for this part is shown in Fig3.4. In order to reach saturation we need at least about 50000 particles. to make the plot (Fig3.4) I ran the simulation 10 times with 800 layer deposits

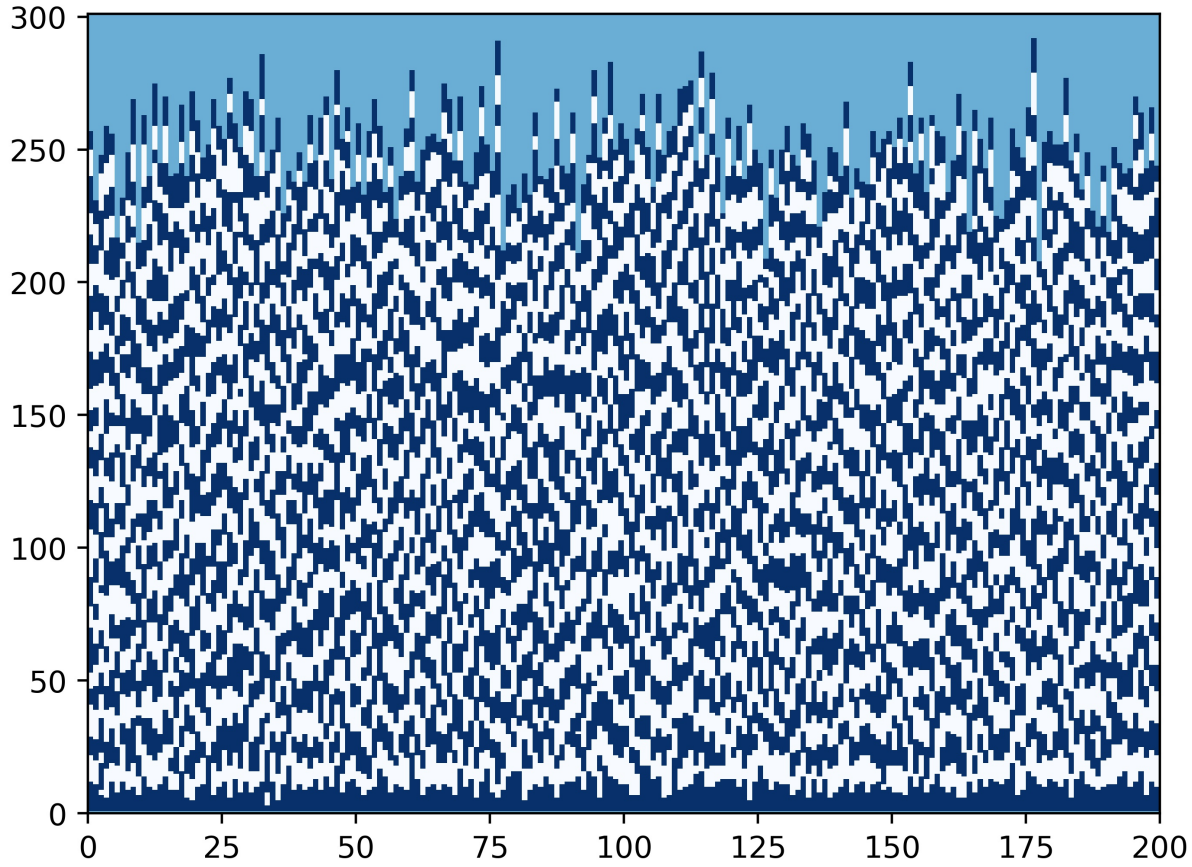

Figure 4: Randomly deposit particles on the surface of length 200 and change color every 2000 particles you deposit. Do this for 50000 particles

total. The value for t_s was output by looking at the graph and getting the data that was closest to the values we were looking for. The value found for beta is:

$$\beta = 0.44 \pm 0.02$$

The values for length L, saturation width w_s and saturation time t_s is as follows:

$$\begin{aligned} L &= [10, 50, 100, 150, 200, 250] \\ t_s &= [64, 1024, 3072, 4096, 8192, 32768] \\ w_s &= [1.5, 4.40, 5.5, 7.2, 7.5, 9.0] \end{aligned}$$

These values are plotted using the file `plots.py`. Plot fittings also happen there.
Values found for *beta* and z are as follows:

$$\beta = 0.30 \pm 0.04, \quad z = 1.75 \pm 0.17 \Rightarrow \alpha \simeq 0.521$$

plots for *beta* and z are found in Fig3.4

The theoretical value for β is 0.21 and alpha, 0.48 which gives a percentile error of $\eta_\beta = 43\%$ and $\eta_\alpha = 8.5\%$ respectively

3.5 Competitive Ballistic Deposition

The dynamics of this system is very similar to that of BD. The canvas plot is shown in Fig3.5

The plot for the Distance is available in Fig3.5. I generated the canvas 10 times and got the mean and sdtev and error for plot.

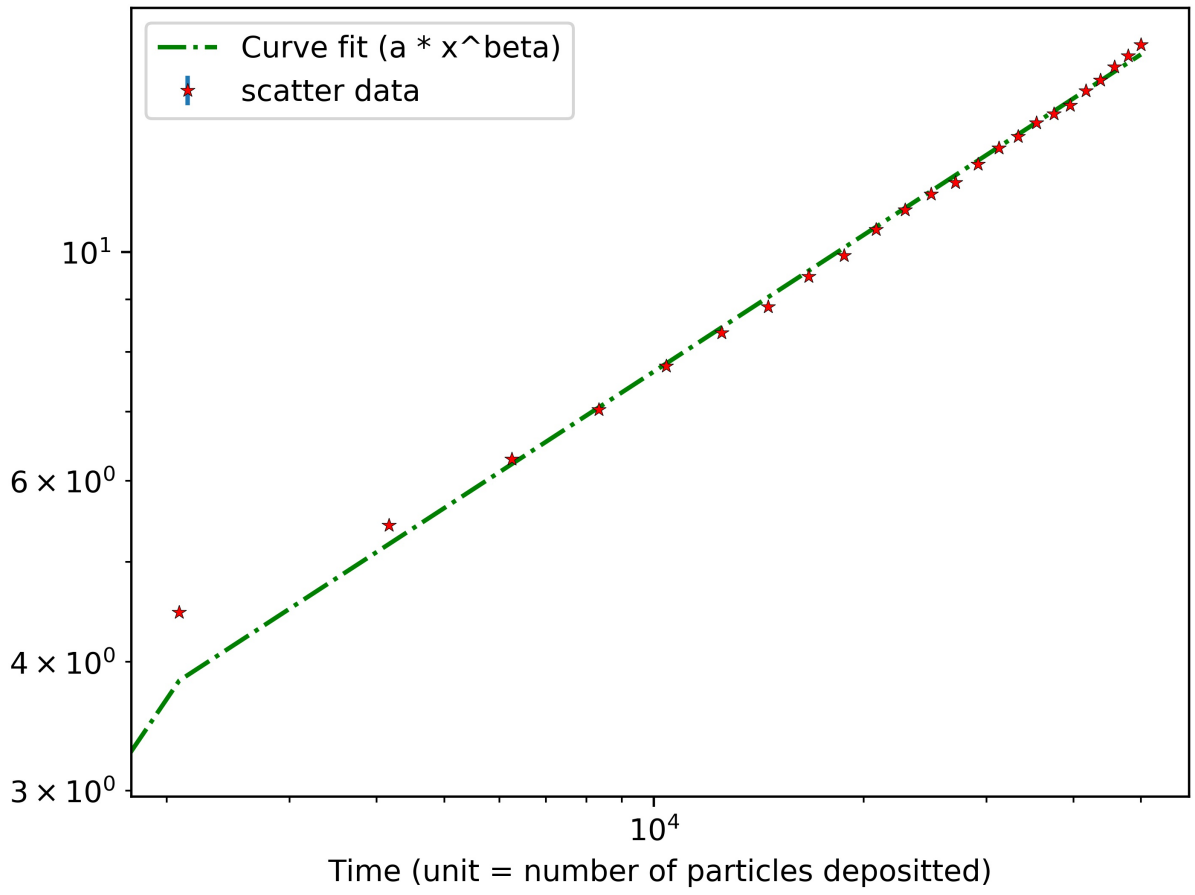

Figure 5: Take the stdev of height as roughness ($w(L,t)$) for every layer you deposit. do this 10 times and find the error of each layer. Then plot and fit using `plt.errorbar` and `scipy.optimize.curve_fit` respectively.

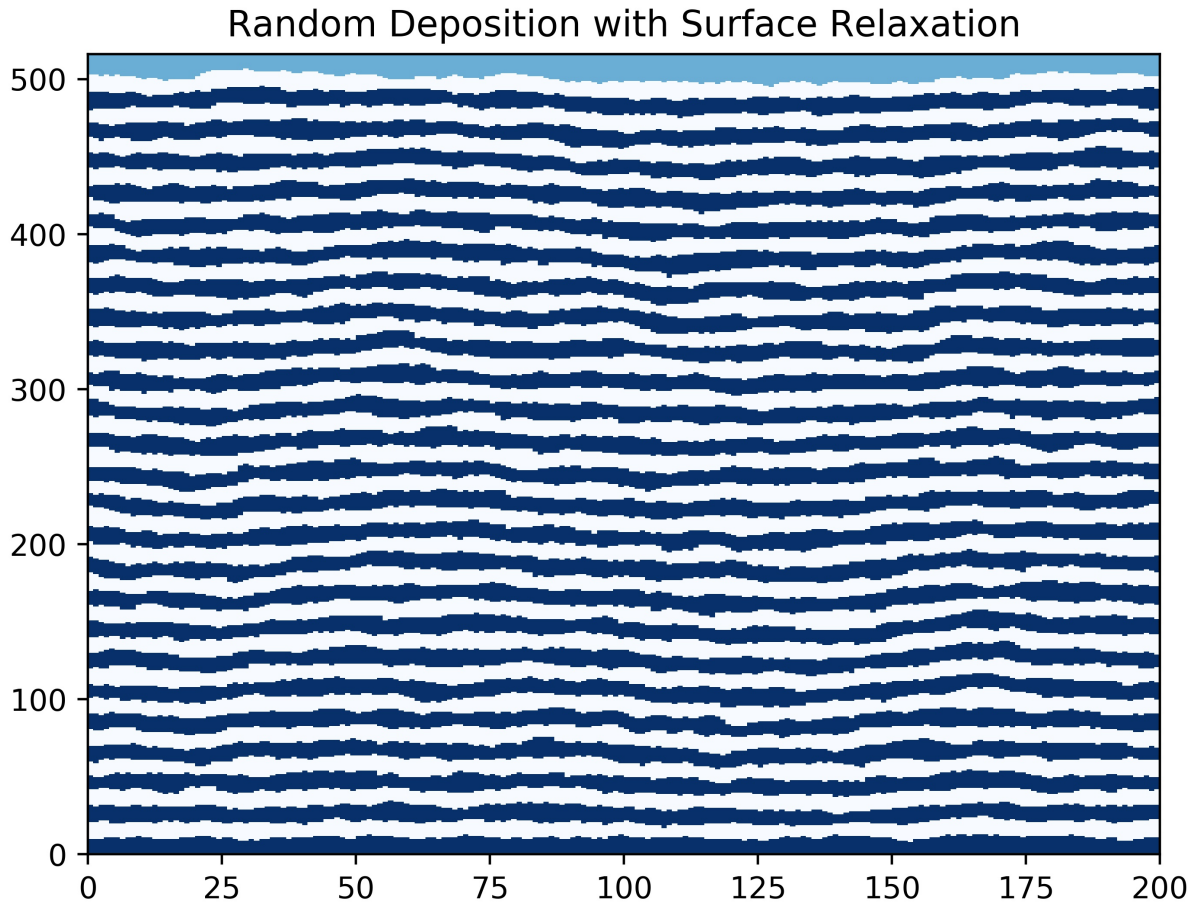

Figure 6: deposit 100000 particles using the RDSR rule and output the data.

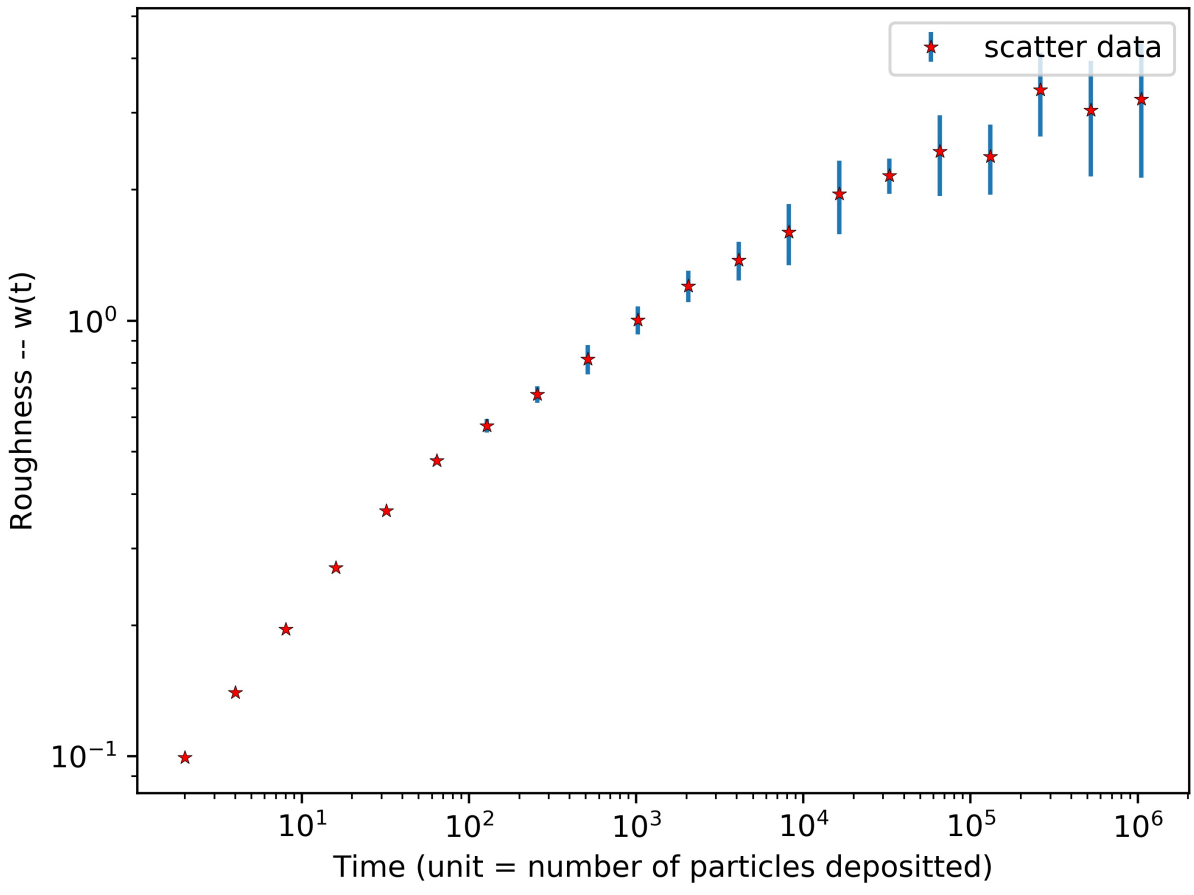

Figure 7: Plot log-log of roughness over time for RDSR

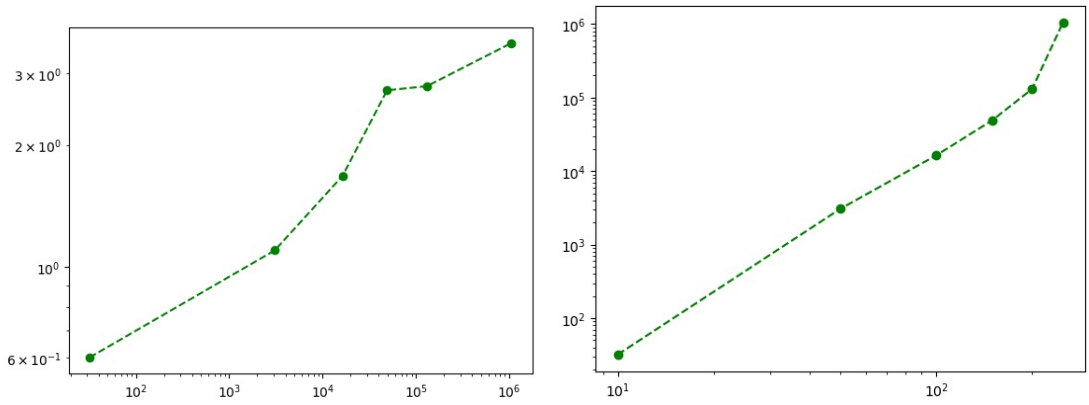

Figure 8: The left log-log plot is for w_s over t_s and the one on the right is for t_s over length$L$. These plots are to measure *beta*, z and *alpha* in RDSR

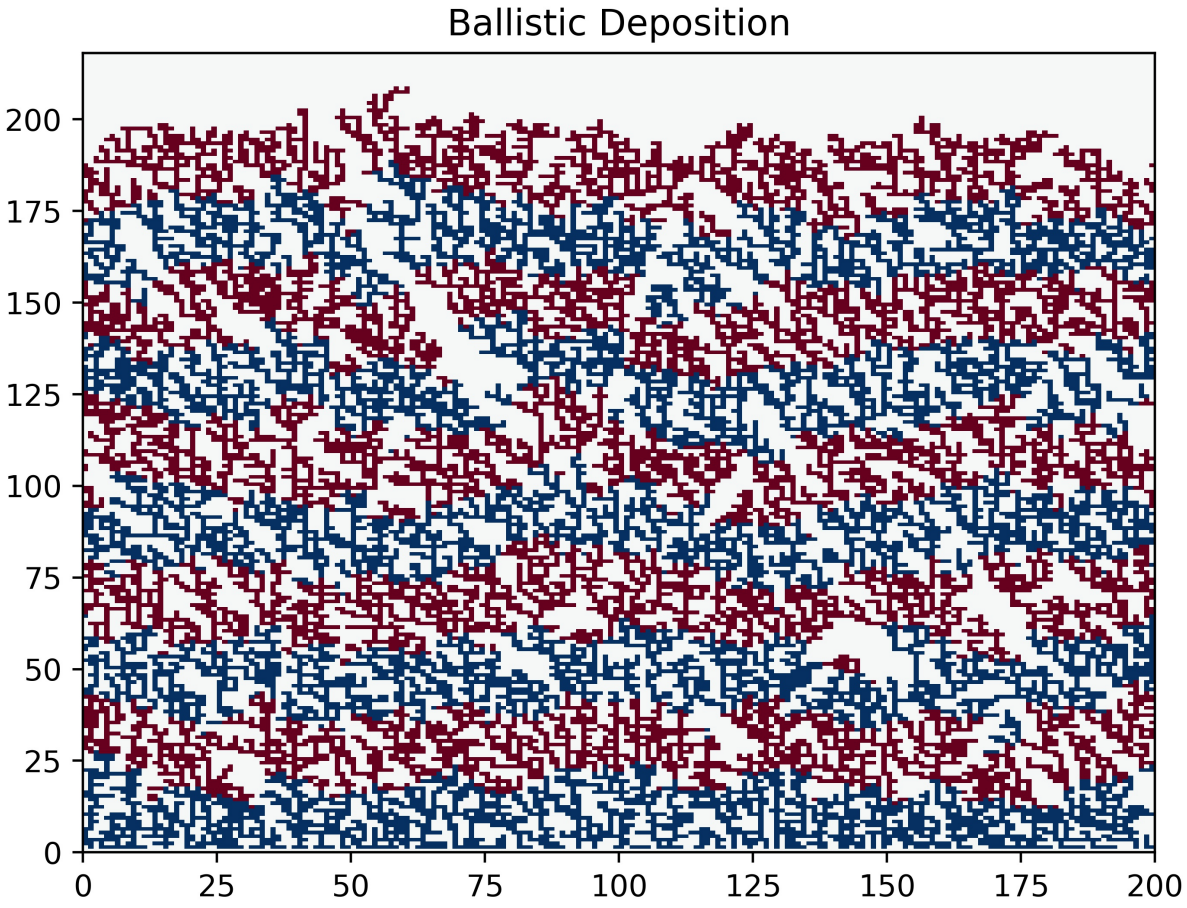

Figure 9: deposit 20000 particles using the BD rule, change colors every 2000 particles and output the data.

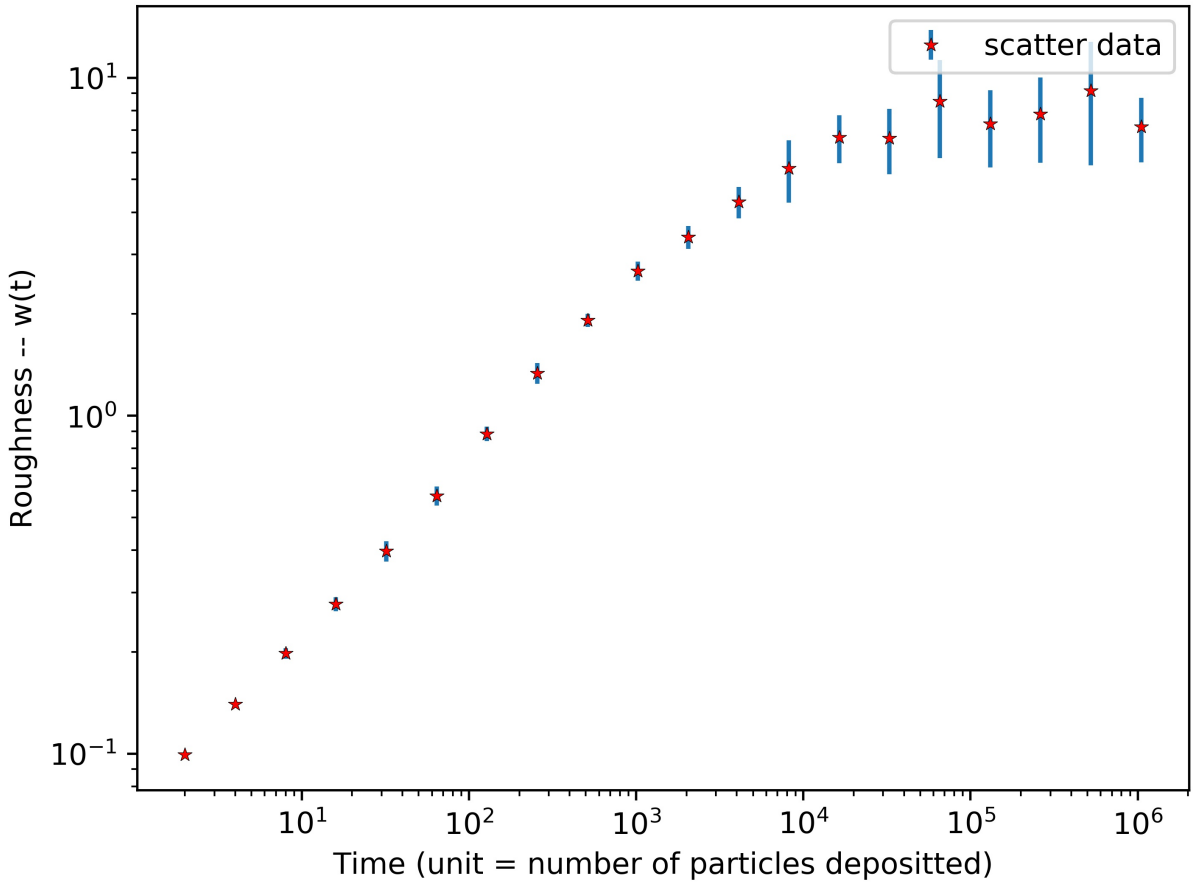

Figure 10: Plot log-log of roughness over time for BD with surface length 200 and 1000 layers laid. Data taken for every 2^n particles laid

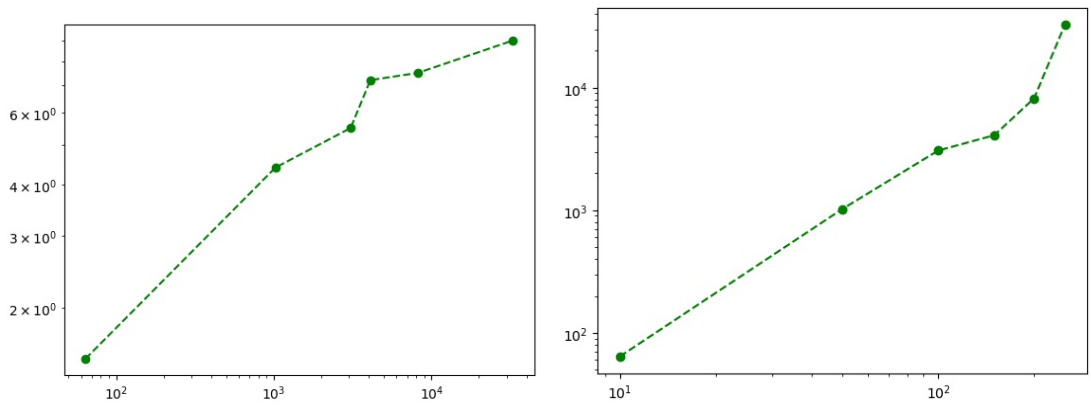

Figure 11: The left log-log plot is for w_s over t_s and the one on the right is for t_s over length$L$. These plots are to measure *beta*, z and *alpha* in BD

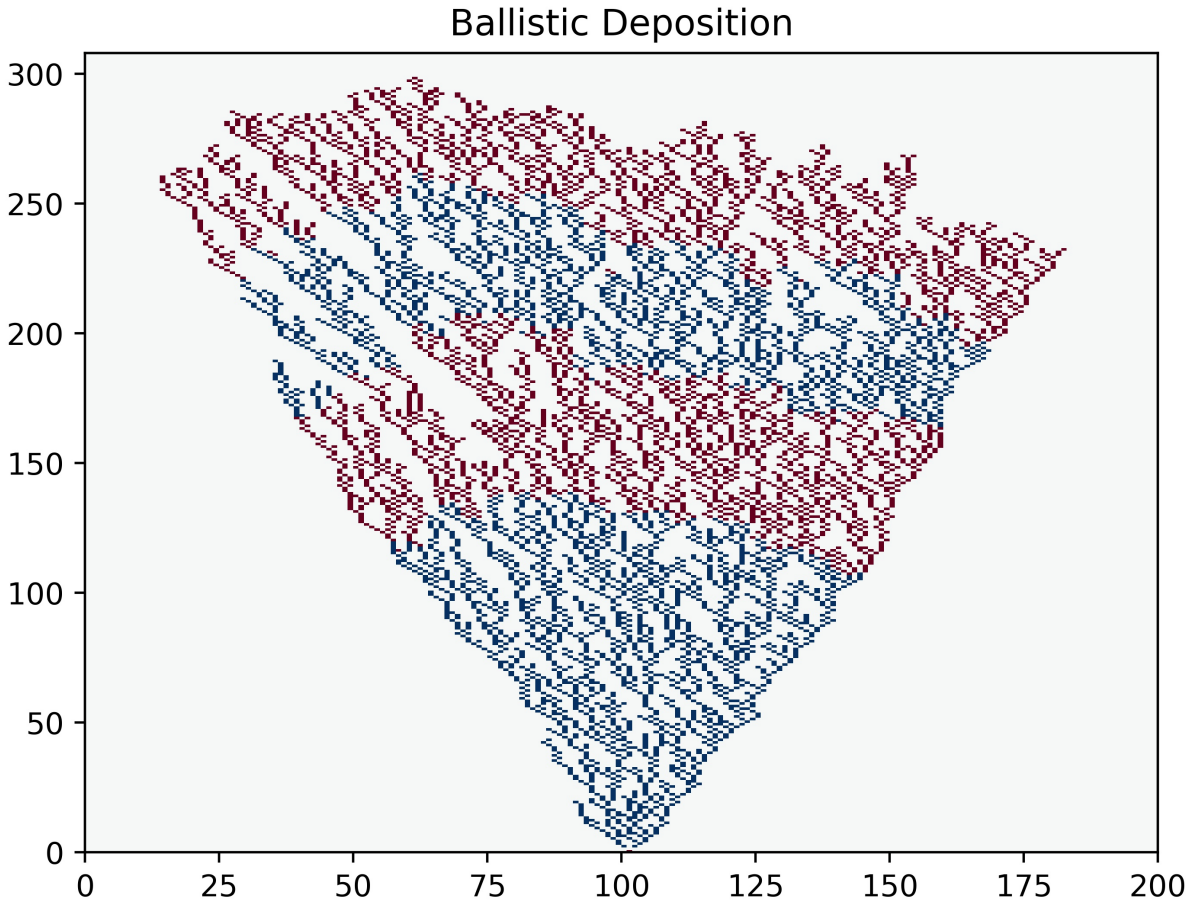

Figure 12: The plot for laying 8000 particles using the CBR rules and initial conditions. Chnage color every 2000 particles.

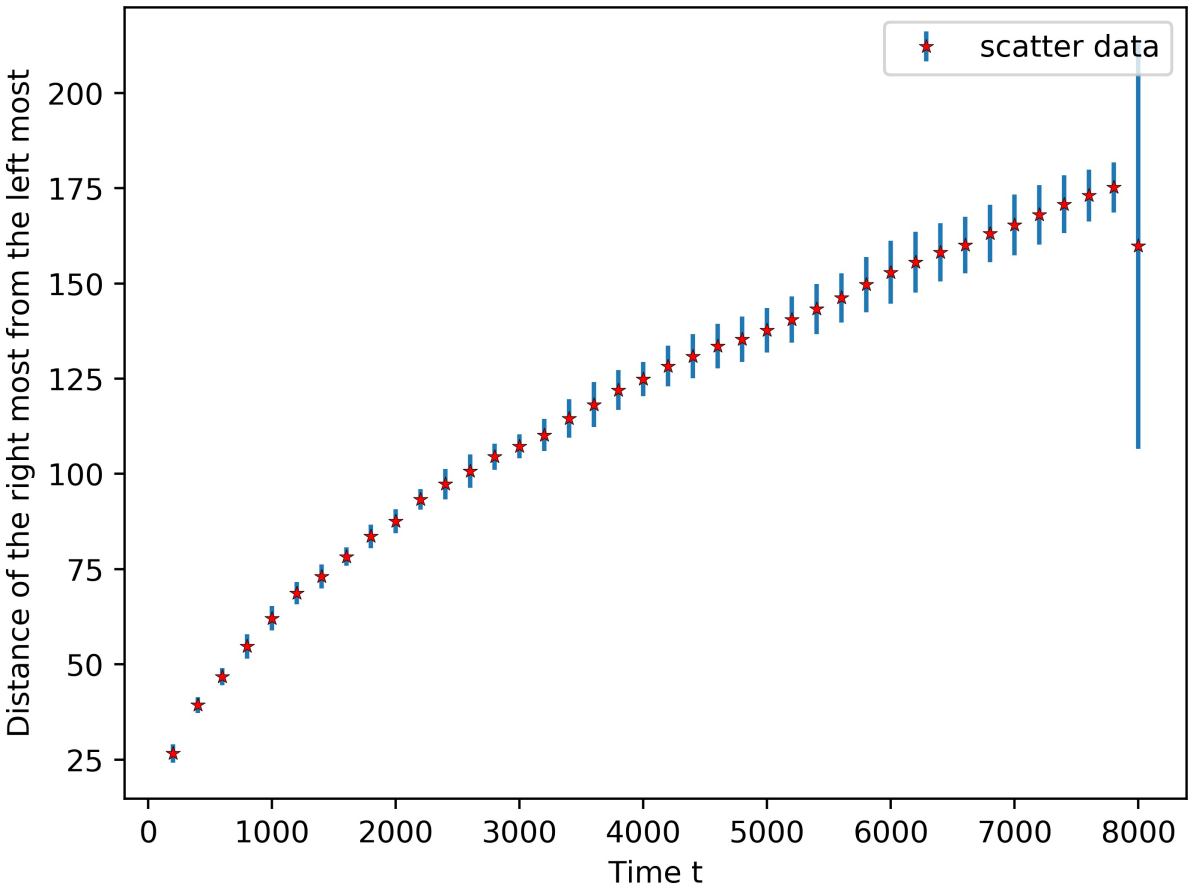

Figure 13: Distance of the index of the right most and left most part of the canvas that is active. over time. Collect data for every 200 particles laid. I generated the canvas 10 times and got the mean and sdtev and error for plot.