Ballistic Deposition

Mahyar Albalan

403201223

1.Random Ballistic Deposition

For this simulation, I used a 200×200 matrix (filled with 0), which represents the sample. Then, I defined a new array called h for storing height information in each column, a random array to determine which column each particle goes into at each step, and another array for calculating the standard deviation at each step. The resulting figure is:

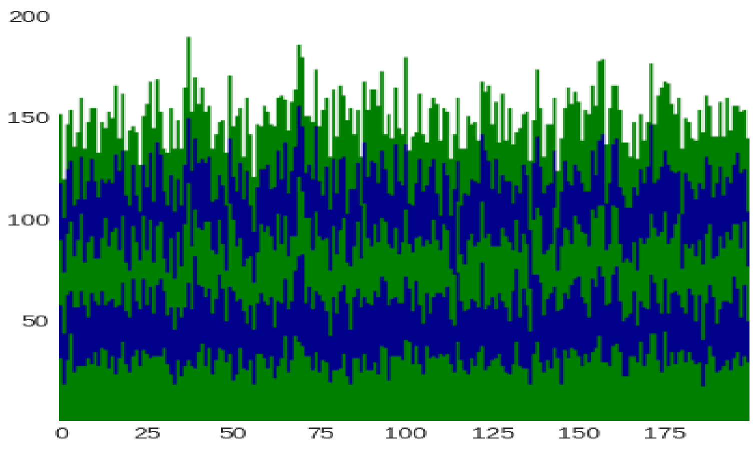
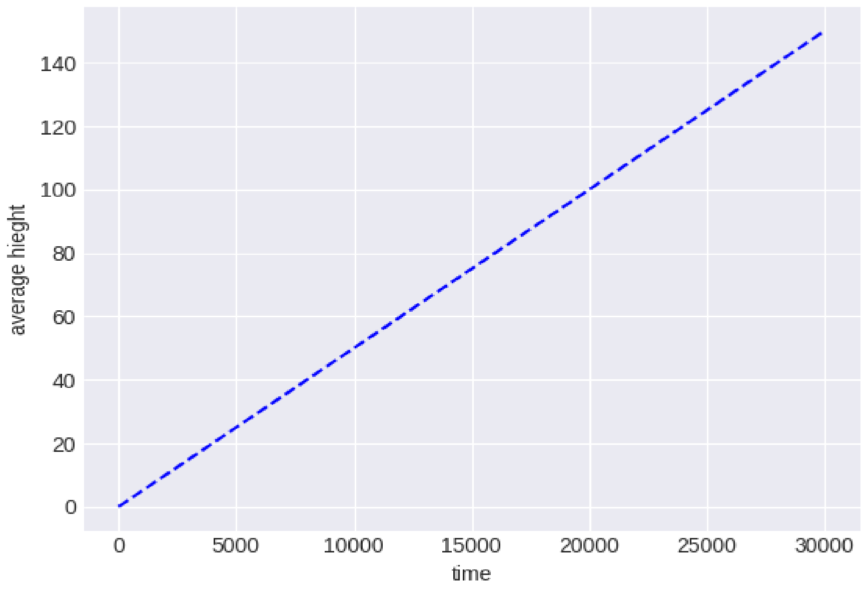


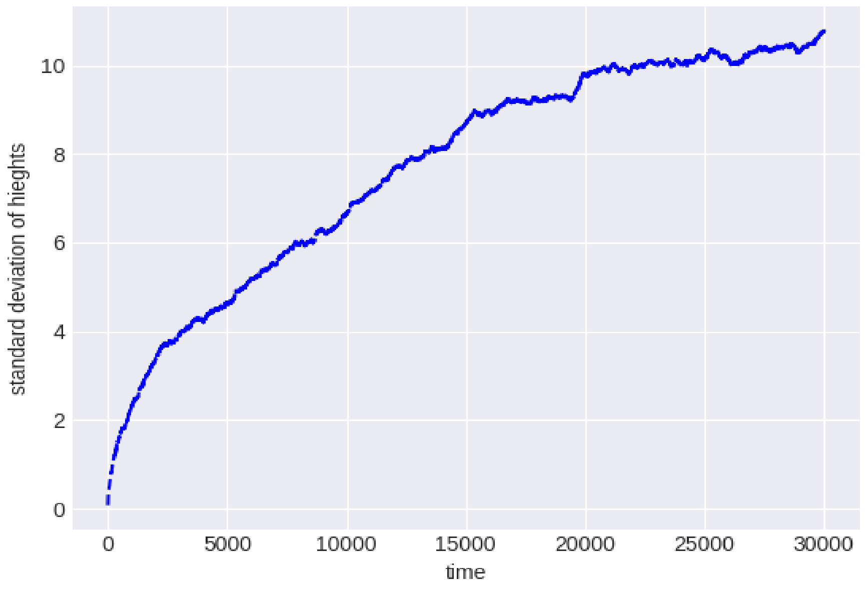
Figure 1: Resulting figure after 30000 steps

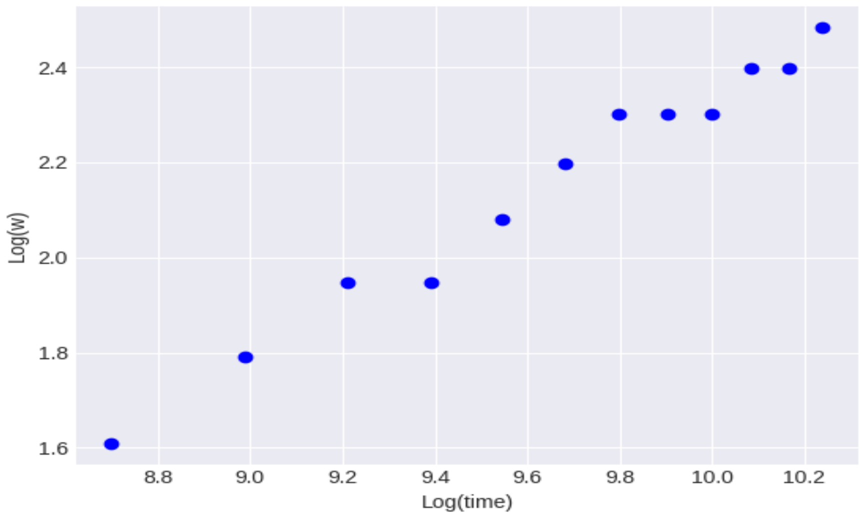
I used green and blue colors for every 30×L (the dimension of the system) steps to show the dynamics of the growth.

Calculating the average height is fairly simple. At each step, one block is added to the matrix, and there are 200 columns in total. Hence, the average height increase at each step would be +1/200. Therefore, the average height plot is linear.



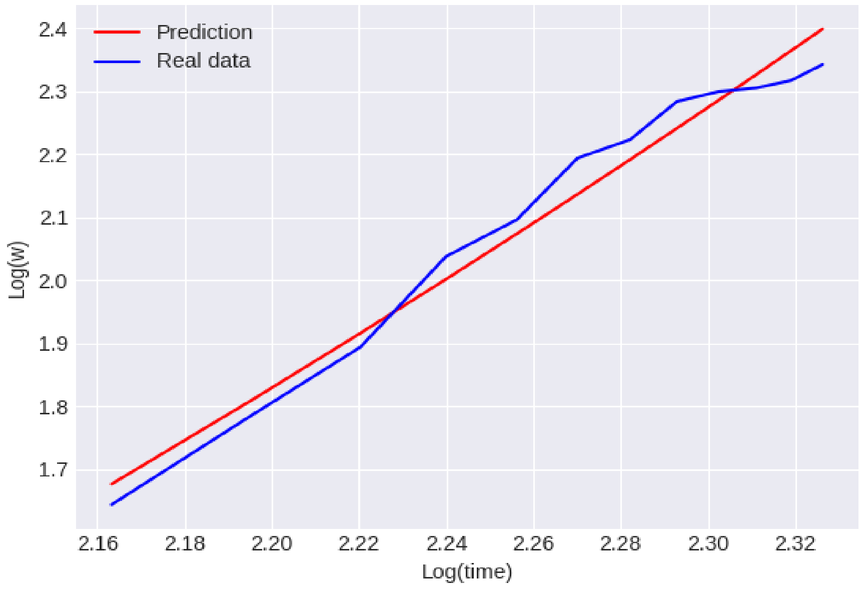
Now, I plot the standard deviation (W) of column heights at each step. To check for a linear relationship, I will plot Log(W) versus Log(time/step).





In the second plot, I do not use all the points—only a subset—to make the linearity more apparent. Additionally, I neglect the first 6000 steps to prevent Log(i) from becoming too large. (This is explained in more detail in the code.)

After this, I used a Linear Regression model that employs MSE as the cost function, as we want, to calculate the best slope for the data. We can see in the plot of the prediction versus the real values that the prediction is appropriate.

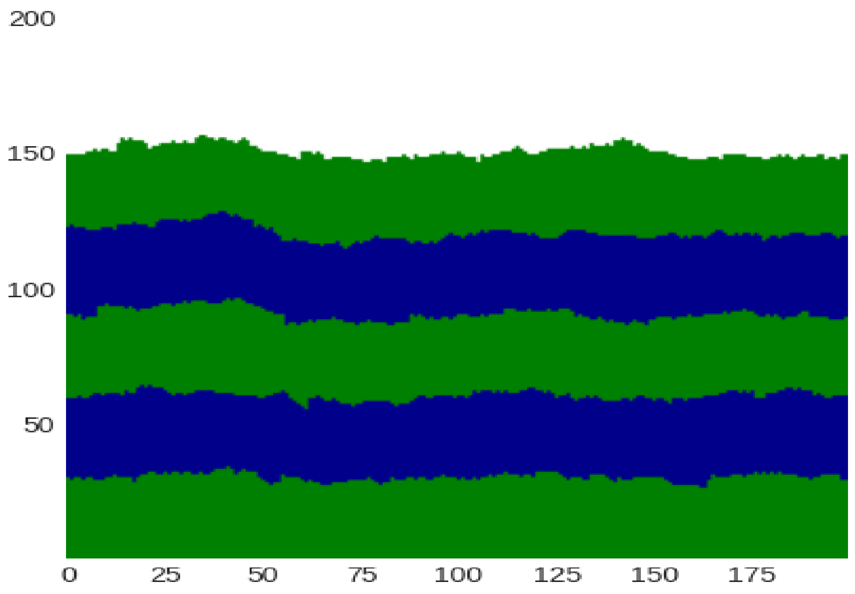


After running this algorithm 100 times, I calculated the average beta and its error as mentioned in the text:

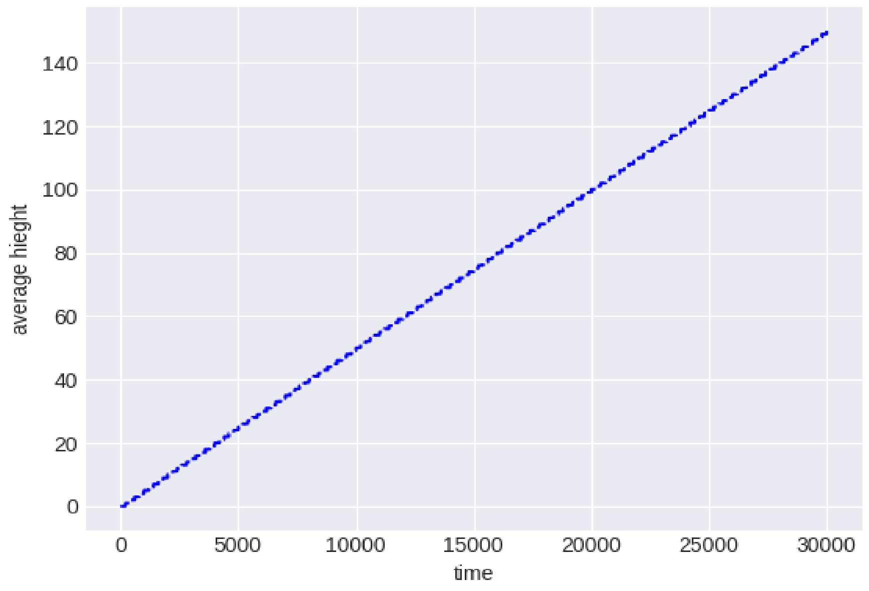
$$\beta = 0.528 \pm 0.047$$

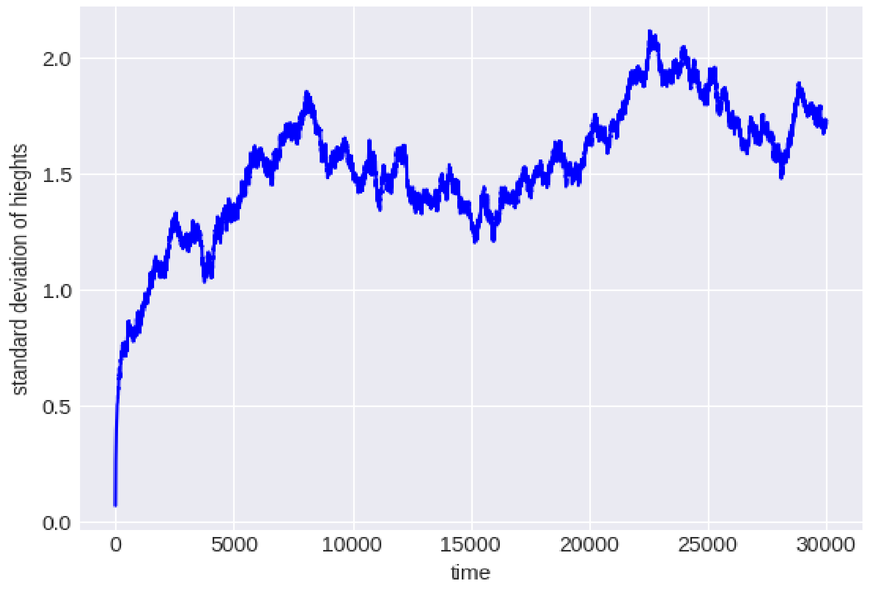
2.Ballistic Deposition with Relaxation

This problem is similar to the previous one; the only difference is that we need a function to check the neighbors' heights, and the particle will fall on the lowest one. *WhereToFill* function will handle this and return the column where the particle should fall. The resulting figure is:

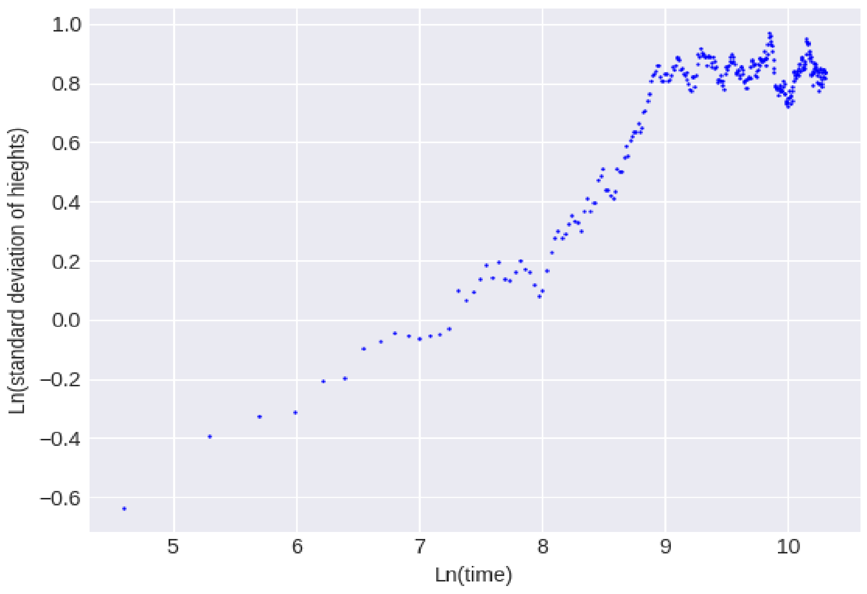


the average hieght and Standard deviation of hieghts are:

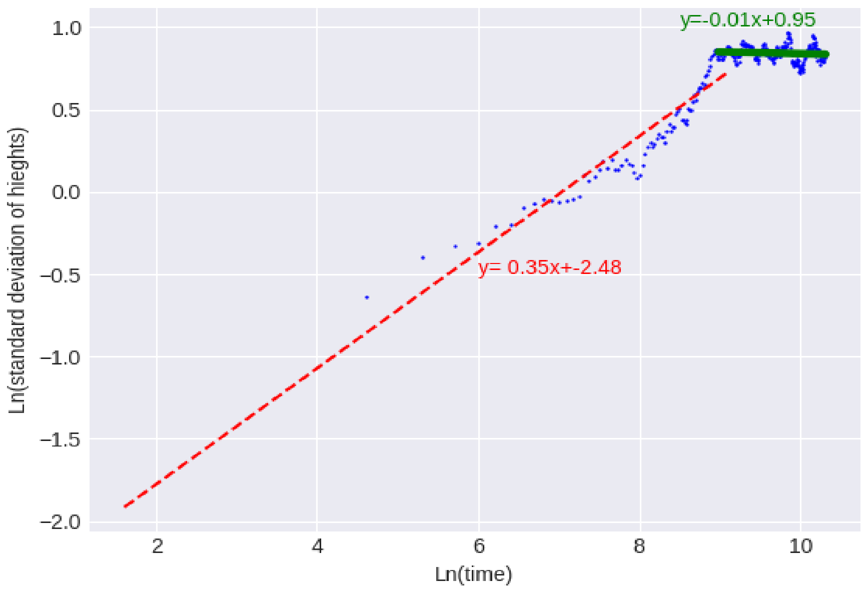




Now i plot Ln(*Standard deviation of hieghts*) versus Ln(*time*):



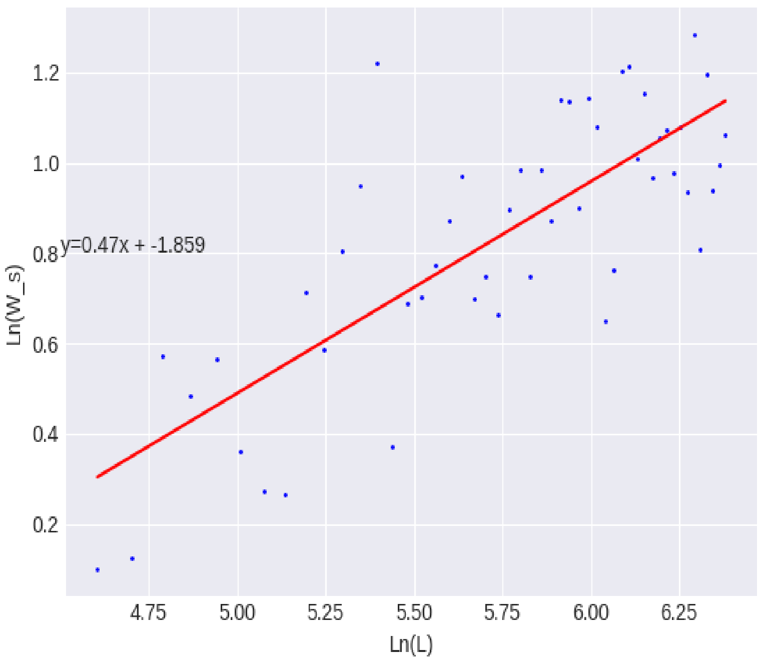
at first as expected you can see the linear realtion but after some times around 8000(after 8000 to 9000 particle deposition) we reach We are reaching saturation point.now i fit two linear models one for linear part and another for after saturation:



I calculated Average of β and its error after doing this process for 50 times:

$$\beta = 0.235 \pm 0.084$$

For Calculating α i will calculate W_s for samples with different sizes,then i use the fact that the α is the slope of the Log(W_s) versus Log(L),first lets plot Log(W_s) versus Log(L):



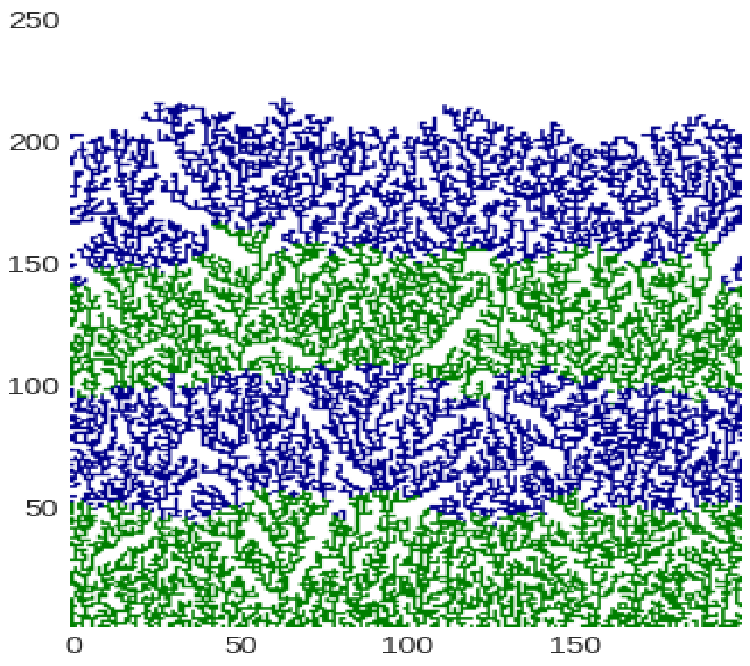
Hence:

$$\alpha \approx 0.47 \Rightarrow z = \alpha/\beta = 1.995$$

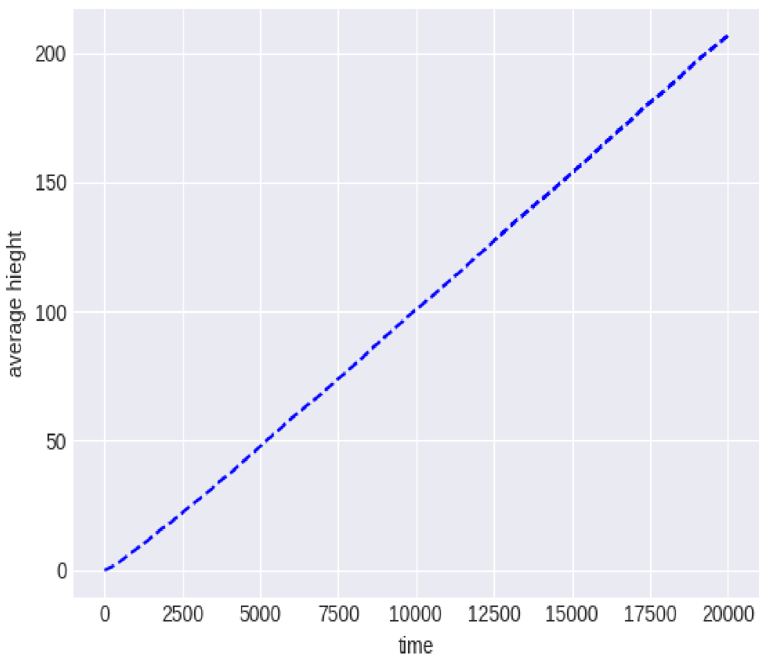
Of course, these results will have some errors. For a better approximation, I could simulate more samples at each L to improve the estimation of W_s. Additionally, in this case, I used 50 different sample sizes ranging from 100 to 600. A more accurate result could be achieved by considering a larger number of samples with higher sizes. The theoretical value for α is 0.433 and my approximation is close, but we can definitely improve it.

3.Side Sticking Model

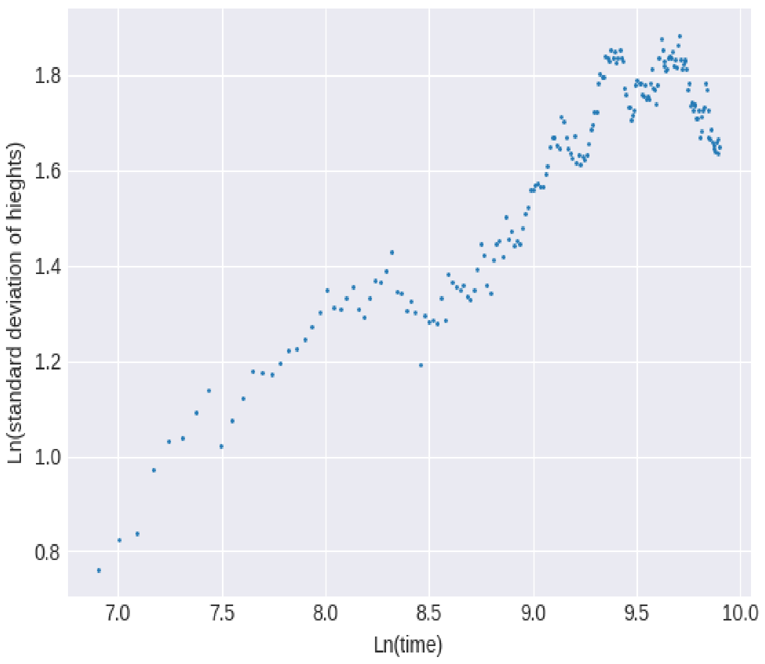
Here, the particle does not move in the x-direction but sticks near the highest neighboring column if the current column it is falling into is not the highest. In the code, the *WhereToFill* function handles this process, while the rest of the simulation follows a similar approach to the two previous models. Here is the result of this simulation:



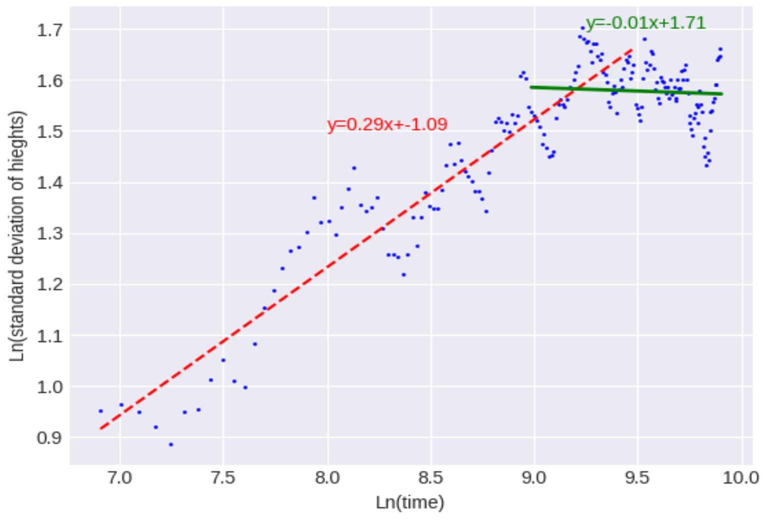
Average hieght in this model still has linear relation with time:



The standard deviation of heights is as follows:



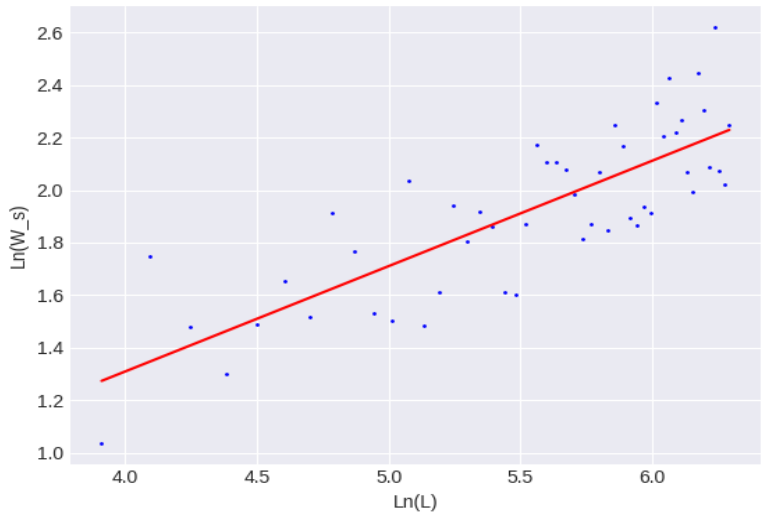
Here again we can see the saturation time is between 12000 and 14000.lets fit linear models to each part as before:



I do this process 100 times to calculate β and its error:

$$\beta = 0.301 \pm 0.0601$$

theorical value is around 0.333, I have around 10% error.
for calculating α and Z, I ran this algorithm for 50 samples with sizes ranging between 50 and 550:



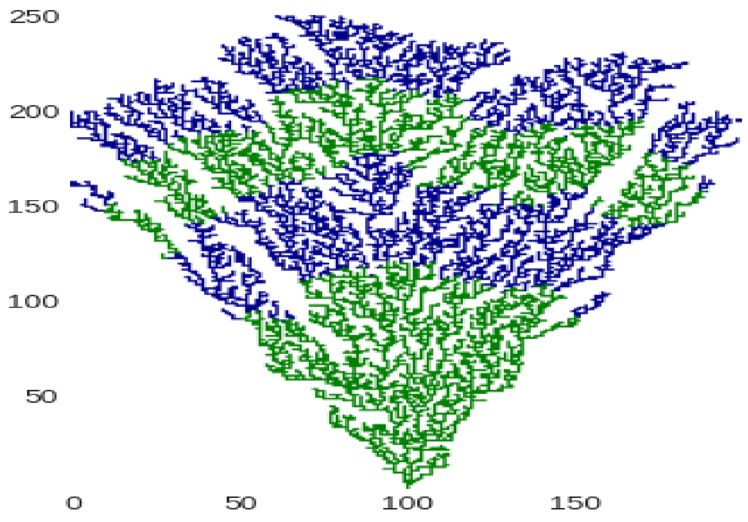
$$\alpha \approx 0.384$$

$$z \approx 1.157$$

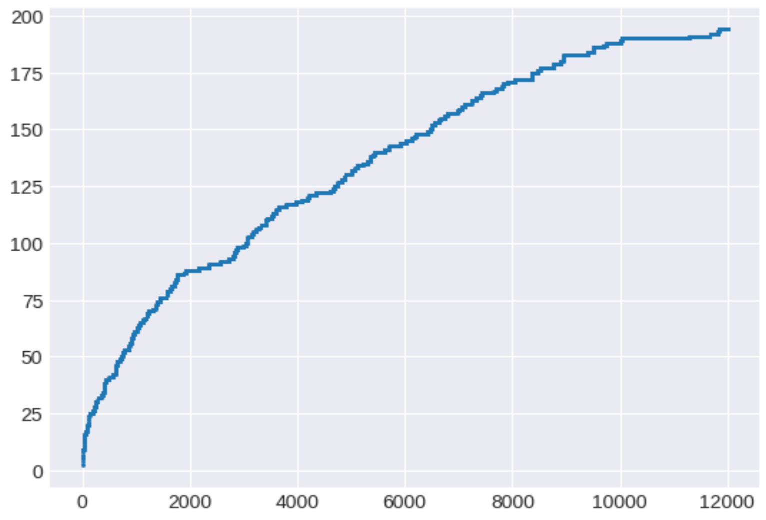
Again, we could have generated multiple samples at each size (here, I used only one for each size, and the total run time was about 20 minutes) to improve these results.the real values for α and Z are,0.47 and 1.75,So I have around 19% error for α and 33% error for Z!.

Correlation length in Side sticking model on single point

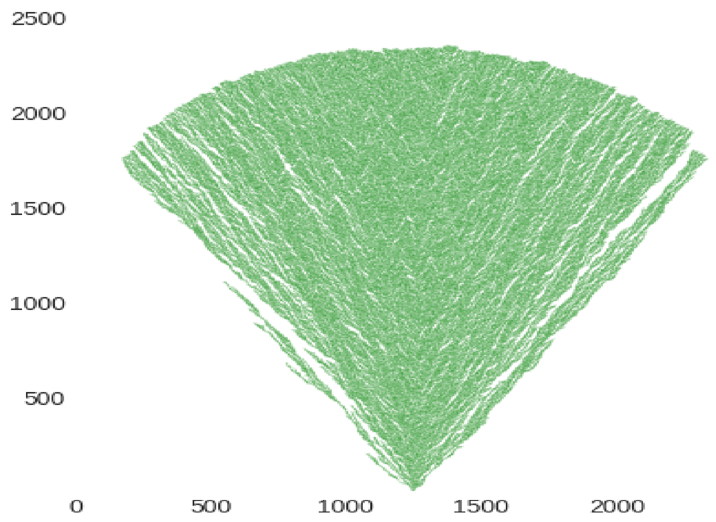
for this model first challenge was that the random generator whould generate so many particles that i should delete them,by this i mean,for example at the start when we have only a single point only 1 column is available and probablity of choosing that column is 1/200 which is very low and again after first particle deposition next deposition probablity whould be 3/200 and so on... for improving the simulation performance i create a function that gives random column but only columns that are available for particles!after adding this function ,*RandomRange*(you can check it details in the code),the proccess was quiet similar to the perevious question, and here is the result for this simulation:

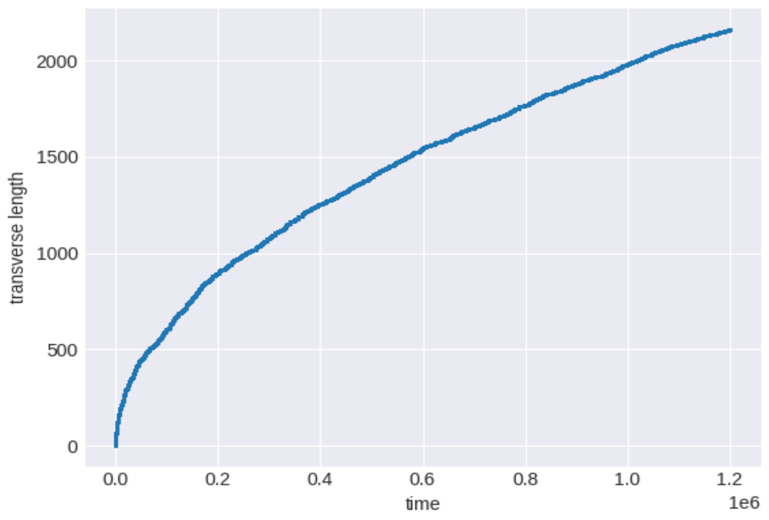


now lets plot transverse length versus time:

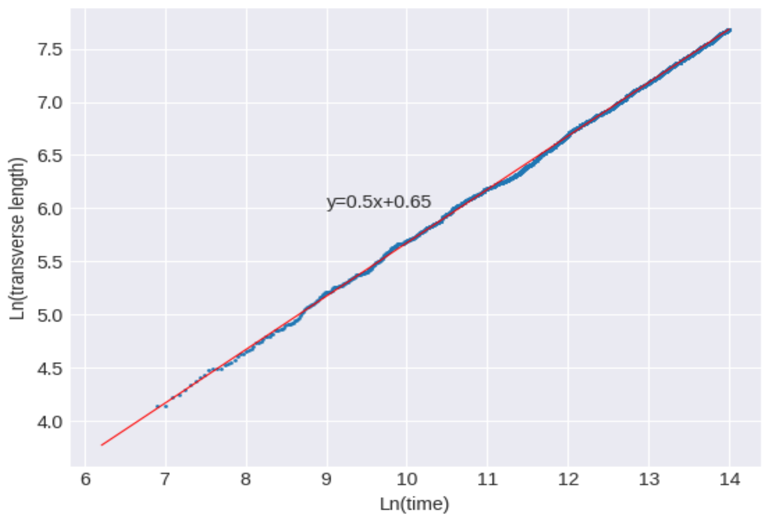


We can see from this plot that the transverse length growth is not linear. However, the number of particles is fairly small (only 12,000). I will run this simulation again for 1,200,000 particles and analyze the result as well.





Again, as you can see, it's not quite linear (it behaves more like $x^{1/2}$), but after around 0.2×10^6, we can observe its linear behavior. Let's plot the logarithmic plot and calculate its growth rate.

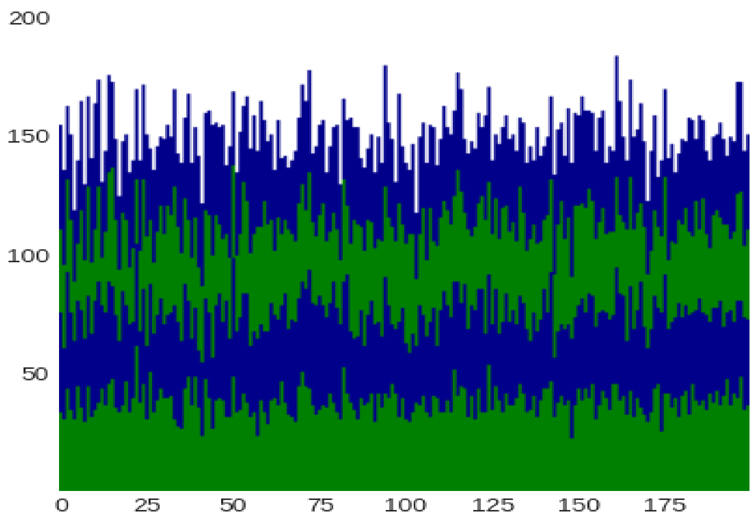


$$GrowthRate = 0.5$$

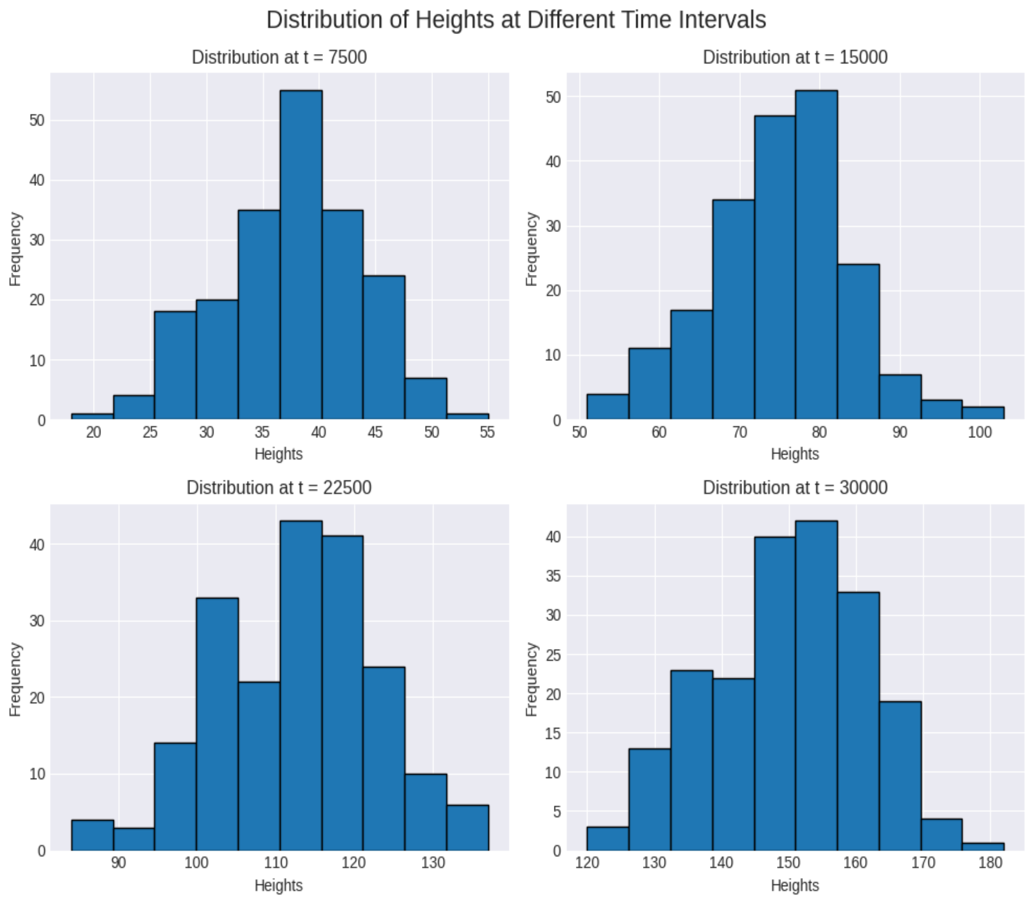
the $t^{0.5}$ growth of the transverse length is consistent with diffusive spreading in deposition processes. This behavior is expected in systems where the active region (where deposition occurs) spreads out randomly over time.so simulation results are correct.

Needle growth

This model is almost similar to Random Ballistic Deposition, except that here, particles drop with an angle θ. For calculating where each particle lands, I wrote a function, *WhereItHit* (check the code for more details). In this function, the particle's trajectory is created first, and then at each step, it checks whether the particle hits a column. The rest of the process is quite similar to before. Here is the result:(i do simulation for $\theta = \pi/4$,But in code you can do it for arbitrary angle)



It is similar to the result of Random Ballistic Deposition (for $\theta = 0$, it is the same). However, here we can see the needle-like structure. This is because the higher columns can absorb more particles; hence, they grow larger and larger.
Here is the distribution of heights at four different intervals:


Distribution of Heights at Different Time Intervals
Distribution at t = 7500
Distribution at t = 15000
Distribution at t = 22500
Distribution at t = 30000
Frequency
Heights