1 Relaxed Random Ballistic Deposition

There is only a small difference between the relaxed, and totally random ballistic deposition layering models. So we can use the same code but with extra rules to plot the graphic. We use two files, one for the graphic and the other for the data.

1.1 rbdrelaxationgraphic.jl

1.1.1 deposit() function

This function takes L, the number of dots deposited and an array representing the maximum height of each bin as input, then outputs an array representing the change in the height of the bins. This function deposits dots by comparing the heights of the neighbouring bins and depositing the dots in the bin with the lowest height. if the height of the center bin is equal to one or both of the neighbouring bins, the center bin is selected. And if the neighbouring bins have the same height and are lower than the center bin, one of them is chosen randomly. Also the outermost bins are connected and act as if the system is in a loop.

1.1.2 Plotting

We run the deposit() function four times, which each of them will be represented in a different color and each time the new heights array will be given to the function. Then we use a grouped bar plot from the StatsPlots.jl package to stack the data on top of eachother and plot our graphic.

1.1.3 Results

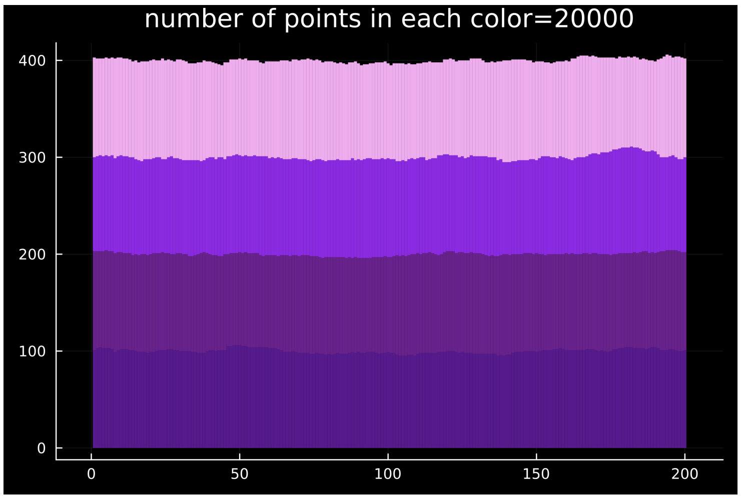

Figure 1: Relaxed rbd for a total of 100000 dots.

1.2 rbdrelaxationdata.jl

1.2.1 deposit() function

This function does the same as the deposit() function in the graphics file with the difference of depositing the dots up until time t all at once.

1.2.2 Hbar() and Hbar2() functions

The Hbar function takes the array H representing the heights of the bins as an input and calculates the average of the heights. The Hbar2() function does the same but calculates the average of height to the power of two.

1.2.3 WGen() function

This function takes the heights array, H as input and outputs W calculated by the formula below:

$$W = \sqrt{\bar{h^2} - \bar{h}^2}$$

1.2.4 Averaging Loop

We define a loop that runs the deposit function multiple times and for different t, then averages the *Barh* and W values and puts them into arrays corresponding to the time t. This loop is not efficient and this cause me to not be able to deposit enough dots or run the simulation enough times to create a good average.

1.2.5 Plotting, Fitting and Results

We use a standard scatter plot to draw our data. We also use the Polynomials.jl package to fit a line to the curve and we also draw this curve on top of our data.

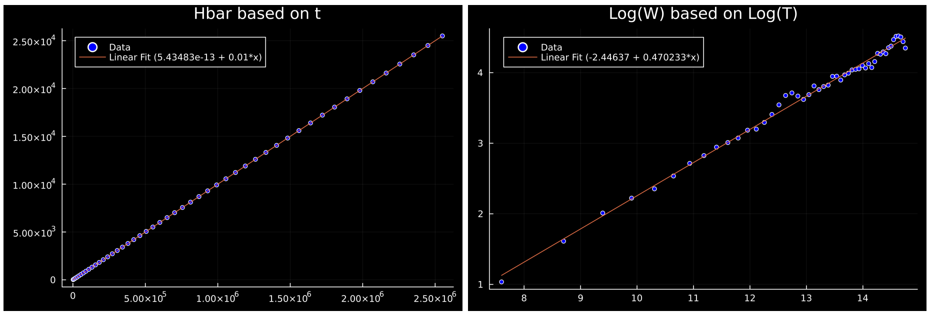

Figure 2: Results for the relaxed ballistic deposition.

2 Relaxed Ballistics Deposition with Side Sticking

The overall structure of the code of this section is similar to the code of the previous section so we will go into less detail.

2.1 rbdsidegraphic.jl

2.1.1 deposit() function

This function takes the number of bins, the number of dots to deposit and an array representing the maximum height of each bin as input, then by abiding the game's rules returns two arrays representing the x and y coordinates of each deposited dot and an array representing the maximum height of each bin.

2.1.2 Plotting

We run the deposit() function four times, each time giving the maximum height array of the last one to the next. We store the values of x and y of each function call in different arrays and we plot them using a scatter plot and a marker that is scaled to be exactly 1×1.

2.1.3 Results

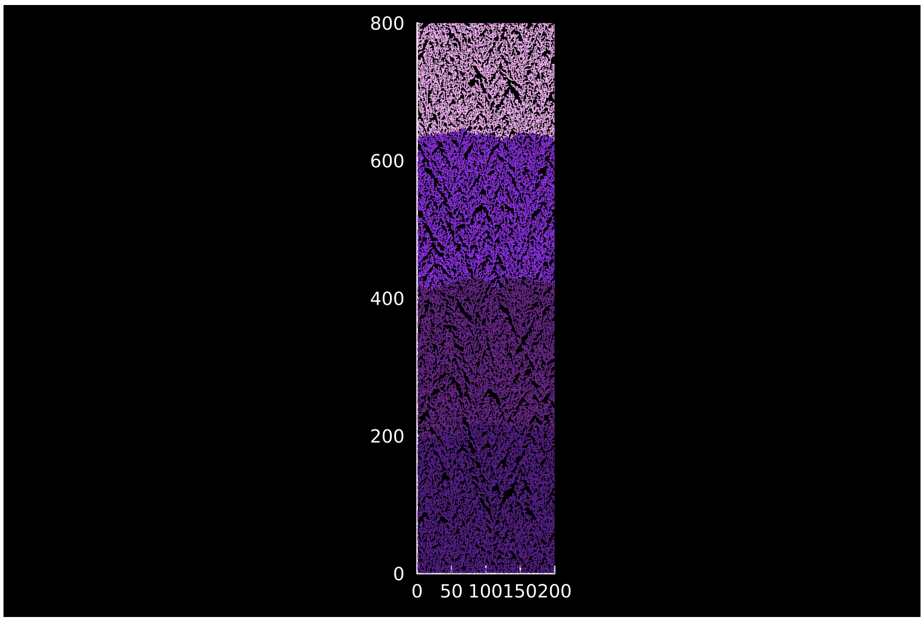

Figure 3: SideSticking rbd for a total of 100000 dots.

2.2 rbdsidedata.jl

The data analysis file is the same as the one in the last problem but with a different deposit function so we won't go into detail but it has the same problems too.

3 Tree Ballistic Deposition

The graphic function for this problem is also similar to the one before with the small difference that we should define a ground where dots that fall to the ground are eliminated from the game.

3.1 rbdtreegraphic.jl

3.1.1 deposit() function

This function is almost the same as the function in the previous problem but we add a rule that dots couldn't stick to the ground where $h = 0$ but we add a dot at height $h = 1$ to start the growing process. the function returns the x and y coordinates of the deposited dots.

3.2 Plotting and Results

To plot the results we use a scatter plot with a marker that is scaled to be $\times 1$. We add the dots of further depositions on top with different colors to create the plot below:

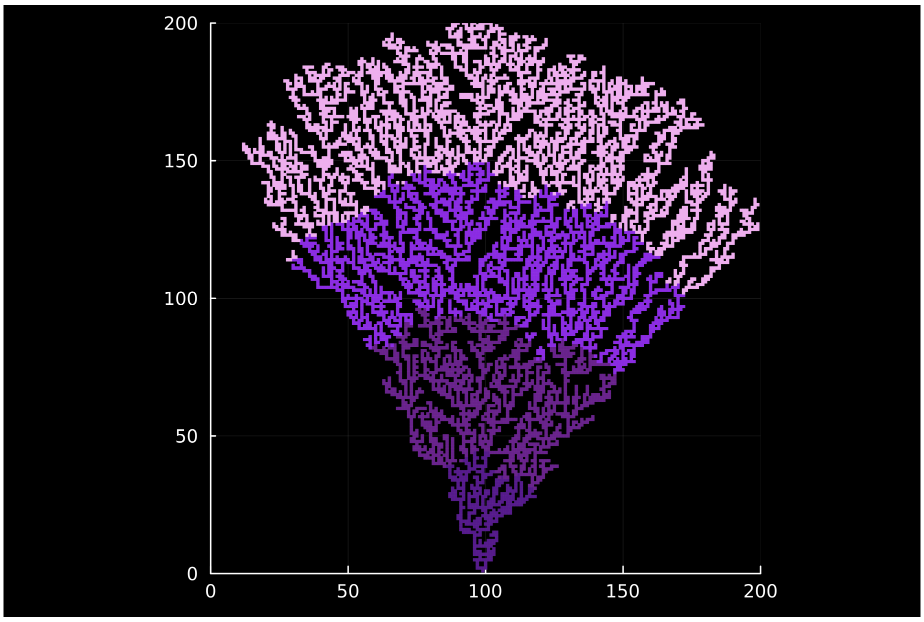

Figure 4: Tree Ballistic Deposition Results for n=20000

3.3 rbdtreedata.jl

We use this file to calculate the with of the resulting tree based on its height.

3.3.1 deposit() function

This is the same function used to deposit the dots in the graphics file.

3.3.2 Width Calculation

We use a loop that iterates on the Y position array for all available heights. Then it searches for the elements with the same value as that height and puts the corresponding X position in another array called row. Then the function calculates the difference between the minimum and maximum amounts stored in row and puts the resulting width in an array.

3.3.3 Plotting and Results

We use a normal scatter plot and plot the widths based on the height of the tree and we are left with the resulting graph: We should also be wary of the

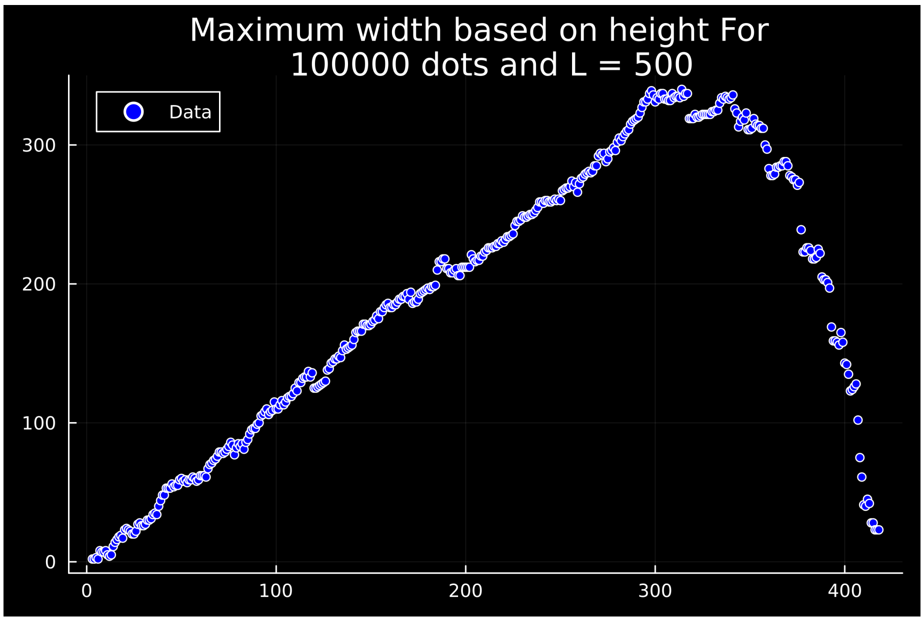

Figure 5: Width of the tree based on its height.

number of bins we have when going to deposit larger amounts of dots since the with stops growing after reaching the limit but if we account for that, the plot scales infinitely.

4 Competitive Ballistic Deposition

In order to create a competitive ballistic deposition model, we send the beams from an angle θ from the horizon. Then from the left we check if the beam intercepts a pillar. the dot will be deposited in the pillar that is hit first.

4.1 deposit() function

This function applies the above instructions. It takes the number of bins, the number of dots and an array corresponding to the height of the bins as an input and returns the array after depositing.

4.2 Plotting and Results

We use a bar plot to show the height of each bin and we are left with the resulting plot:

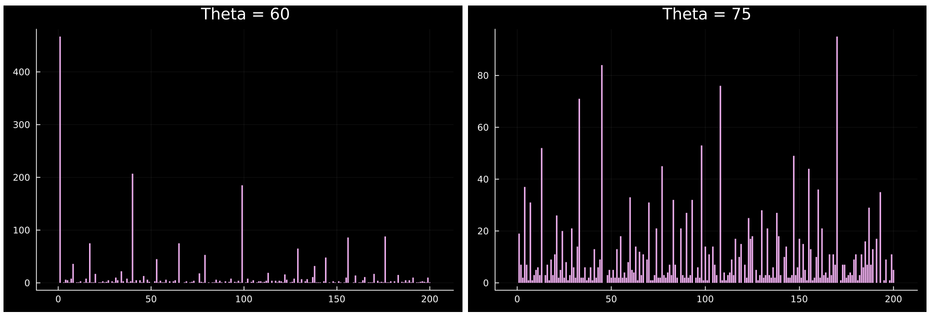

Figure 6: Results for the competitive ballistic deposition.