Complex Physics - Midterm 2

Tomás Ricardo Basile Álvarez

October 5, 2023

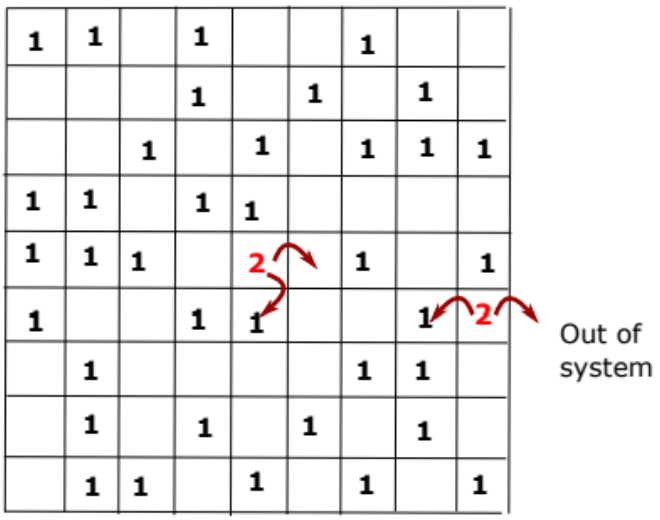

Figure 1: **Lattice with two active sites that each relaxes by moving grains to their neighbors. Sites on the boundary of $L \times L$ system can relax by putting grains out of the system.**

Consider a model on the 2d lattice with size length L in Fig. 1. Each lattice site x, y can take value $h_{x,y} = 0, 1$, or more. Sites with 1 or less grains are relaxed. The system is excited by adding a grain to some random position x, y. Grains are only added after the entire system is relaxed, i.e. all $h_{x,y}$ less than 2. When the system is excited it is relaxed in a sequence of relaxation moves where sites where $h_i > 1$ relax. This is done by moving two grains randomly to the four nearest neighbors of the excited site (grains are moved out of the system when the excited site is on the boundary of the lattice). Each grain is assigned to these new neighbor positions randomly and independently of each other. We recommend synchronous toppling, where all grains with h above 1 topple at the same time (hint: define a current lattice and a lattice of active changes).

Part 1.

Implement the above model for a $L \times L$ lattice with $L = 25$ and plot the sequence of avalanches as a function of time. The avalanche size is defined as the number of single-site relaxations until the new relaxed state is reached. Convince yourself when steady state dynamics is reached (this critical attractor is characterized by avalanches that do not anymore increase in size).

The model implementation using python is shown in the code document and it basically consists of the following steps:

1. Define a function "create-array" that creates an $N \times N$ array of zeroes.
2. Initialize an array called q in which we will save the points that need to be toppled.
3. Define a function "grain" that takes as input a position in the array, adds a grain to it and if the number of grains goes above 1, it adds the point to the array q so that it may be toppled later.
4. Use the function grain to add a grain on a random point in the lattice. If the point ends up with more than 1 grain, it will be added to the list q.
5. Then, do the following steps while q is not empty:
   (a) Take the last element of q and erase it from the list.
   (b) Take two grains away from said point and add them randomly to two neighboring points using the function "grain". If the random direction selected goes outside the lattice, just do nothing and said grain will be lost.

I did the aforementioned procedure for an array of 25×25 points and plotted the size of avalanches for 10000 repetitions of adding one grain randomly in the lattice.

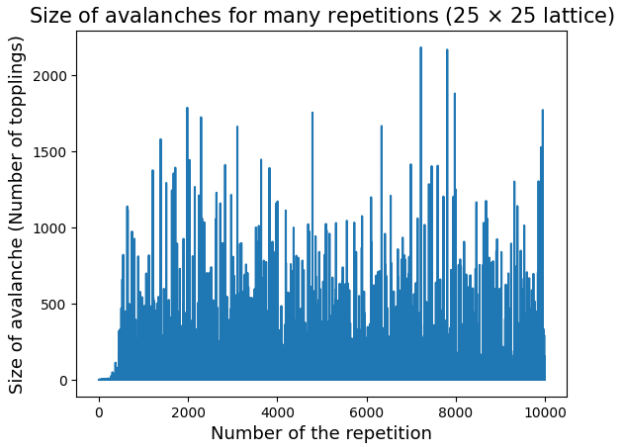

Figure 2: **Avalanche sizes (measured as the total number of topplings) for** 10000 **repetitions of adding one grain to the system for a** 25×25 **lattice.**

We can see from figure 2 that the sizes of the avalanche start small but grow very rapidly and after only about 500 repetitions, they don't seem to grow anymore. Therefore, we can safely say that with a couple thousand iterations (say $10,000$ to be extremely on the safe side), the system reached steady state dynamics.

We can also see the point where we reach the steady state in a slightly different way. To do it, instead of plotting the size of each avalanche, we will plot the cumulative sum of the avalanche sizes. This graph is shown in figure 3.

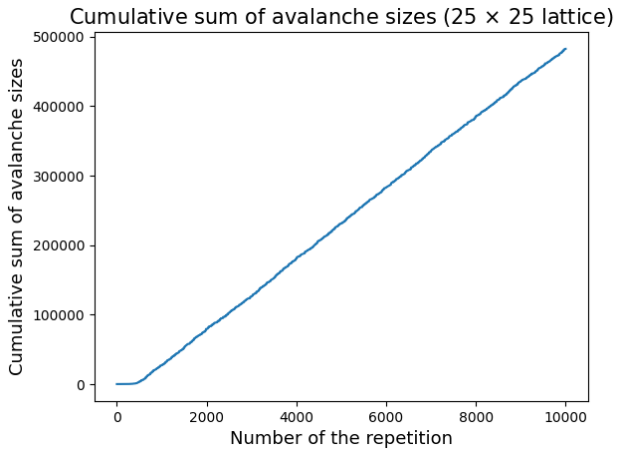

Figure 3: **Cumulative sum of avalanche sizes for many repetitions of adding one grain to the system.**

We can see that at first the cumulative sum does not grow too much, since the first few avalanches are very small. However, after about 500 repetitions, the cumulative sum grows linearly (with slight fluctuations), showing clearly that after said point, all avalanches have nearly the same size and are not growing in size any more.

However, this was the case for a 25×25 lattice, but in this work the biggest lattice we will consider is one of size 250×250, so we will have to know the number of repetitions to get to steady state for this size of lattice. Therefore, I implemented the same algorithm and plotted the same graphs for a 250×250 lattice, but this time with 150000 repetitions. The plots are shown in figure 4.

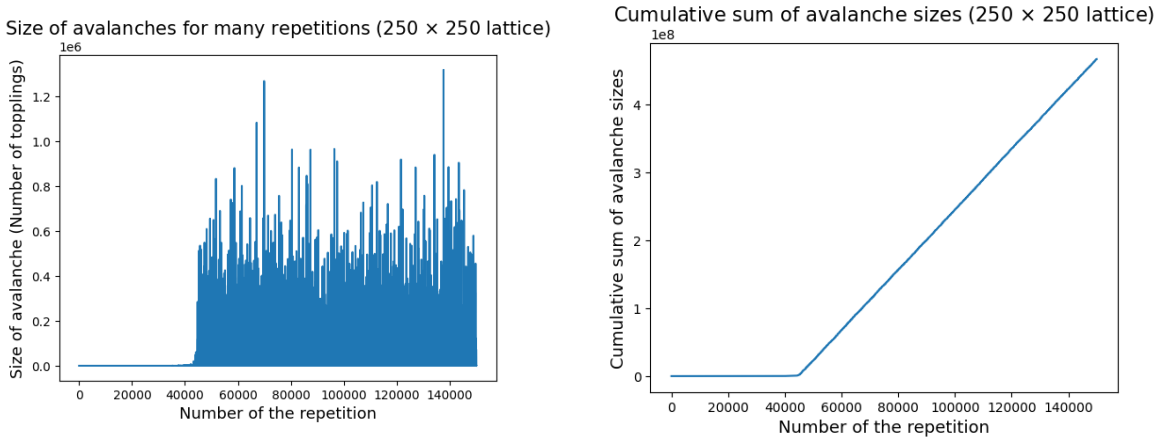

Figure 4: Sizes of avalanches for a 250×250 lattice. On the left side we can see a plot of the sizes themselves and on the right side the cumulative sum of said sizes.

From figure 4, it is clear that after around 45000 iterations, the avalanche sizes stop growing. Therefore, for this big lattice, to be safe, we will use 60000 iterations before considering that the system has reached a steady state. Actually, we will use 60000 warm-up iterations for all lattice sizes, since it doesn't really make a difference for smaller lattices and the program is fast enough so that this doesn't significantly affect the computation time.

Part 2.

Consider system sizes $L = 25, L = 50, L = 100$ and $L = 200$ and simulate each system until a steady state is reached. In steady state, you should extract the size distribution of the avalanches, and estimate the exponent for the size distribution and for the scaling of the avalanche cut off with system size L (dimension of avalanches)

As the instructions say, for this part we simulate the process for many different sizes (to be exact, we do it for a few more sizes than the instructions say, in particular, all ten sizes from 25 to 250 in steps of 25). For each size, we simulate 210000 repetitions of adding one grain, but discard the first 60000 to make sure that we have reached the steady state.

For each lattice size, we register the avalanche sizes in each of the 150000 repetitions after having reached critical behavior (and we ignore avalanches of size 0). Then, we can plot a histogram of the avalanche sizes. For example, here is the histogram for a lattice of 100×100.

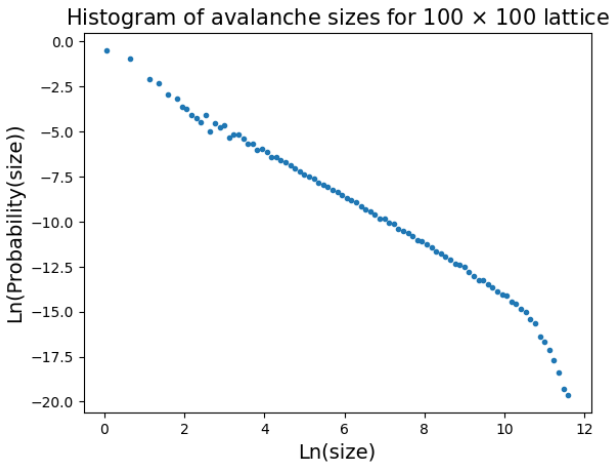

Figure 5: **Histogram of avalanche sizes for a 100×100 lattice. Both axes are logarithmic (natural log).**

We can clearly see that the histogram first follows a linear trend and then decays rapidly. This is to be expected, since we know that the distribution will be of the form

$$P(s) = s^{-\tau} e^{-s/L^D}.$$

Then, taking the logarithm of both sides, we obtain

$$\ln P(s) = -\tau \ln s - \frac{s}{L^D} = -\tau \ln s - \frac{1}{L^D} e^{\ln s} \tag{1}$$

Therefore, on a log-log plot of $P(s)$ vs s, we will get a linear decay with slope τ and an exponential decay characterized by L^D. We can see in the graph that at the beginning the linear decay dominates and then there is a cutoff point where the exponential part becomes

important.

The objective of this part of the assignment is to determine the exponent τ and the term L^D that characterizes the cutoff. There are many ways to achieve this and in particular I did it in two different ways, which I detail below:

1. **Fitting separately:** In this method, I fit separately the linear part and the exponential part to find τ and L^D.

   To do it, I first fitted the first 75% of points with a line using the python function scipy.curve-fit. Then, I plotted said line and found the last few points that fall below it. These points that fall below the line will be the ones that we will fit with the exponential and the remaining points will be fitted with a linear function. Here is a plot of the line fitted to the first 75% of the data points.

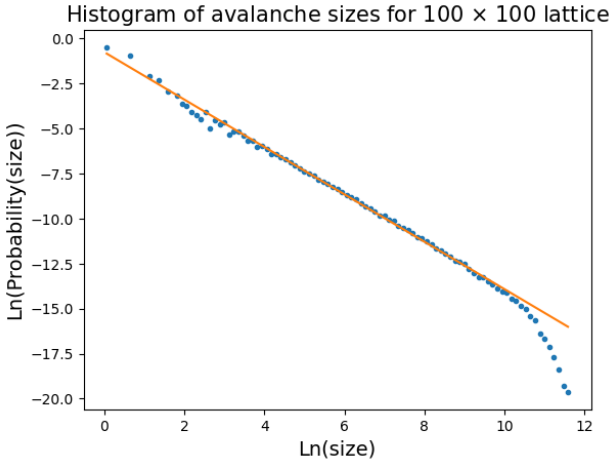

Figure 6: **Histogram of avalanche sizes for a 100×100 lattice fitted with a line for the first 75% of points. The points to the right that fall below the line will be fitted with an exponential, while the rest of the points will be fitted again with a line.**

   As mentioned before, the points below this first line will be fitted with an exponential of the form $ae^{(-\ln s/L^D)}$ so that we can determine L^D. Meanwhile, the remaining points will be fitted with a line $-\tau \ln s + b$ so that we can determine τ. In the following figures we can see the histograms along with the fitting of the linear and exponential parts (though the linear fit done to decide which points to fit exponentially and which points to fit linearly is not shown, since it is just an intermediate step). As I said before, I made this process for 10 lattice sizes from 25 to 250, but here I only show 6 of them.

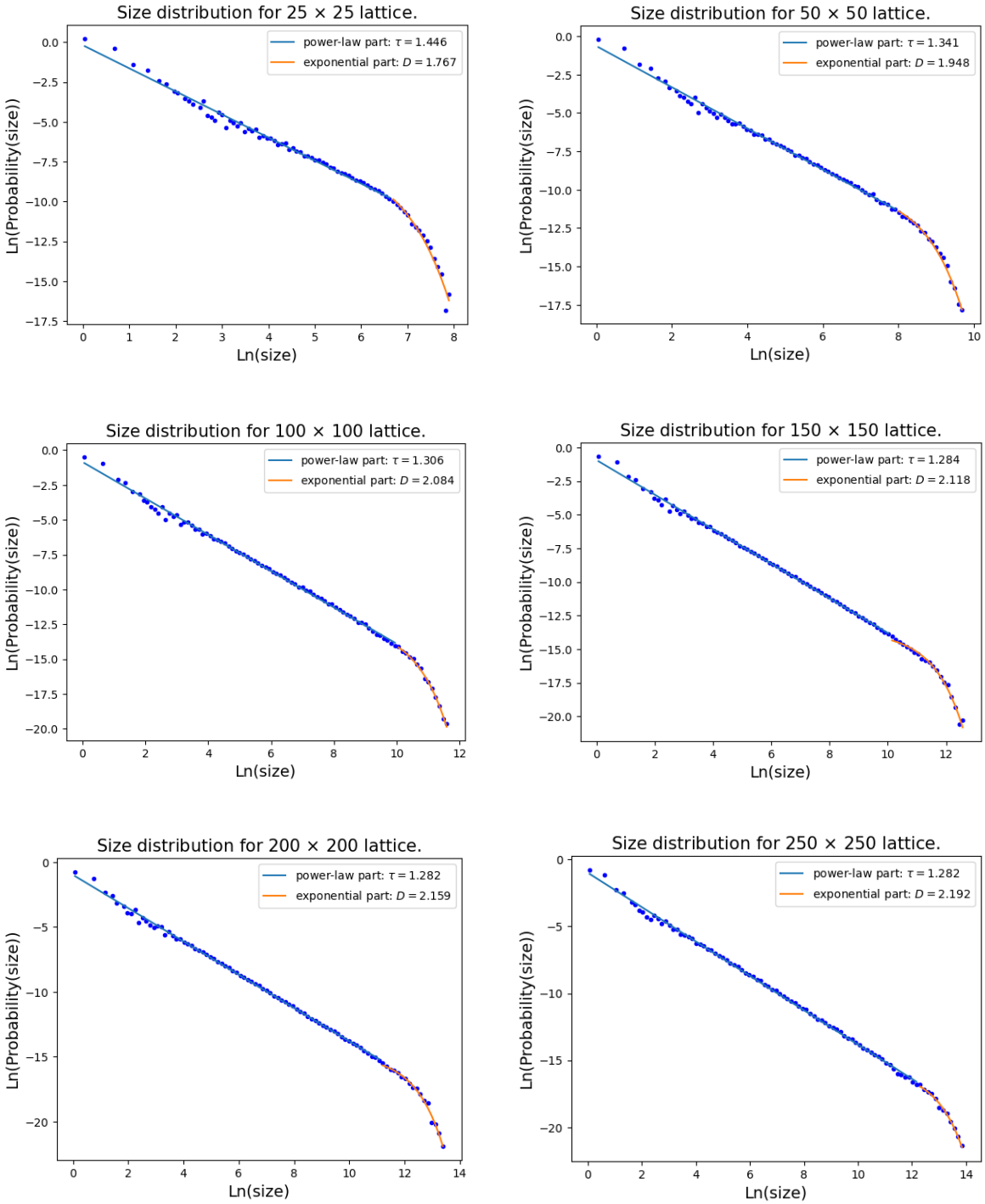

Figure 7: Size distribution plots for many lattice sizes along with the linear and exponential fit and the values of τ and D obtained with the fits.

Given these results, we can now see how the values of τ change with the lattice size. In particular, here I plot the value of τ as a function of $1/L$ (with L the lattice size) and fit it with a line to get the value it tends to as $1/L$ becomes 0. I also do the same

thing for D. For these two plots I used all the lattice sizes I worked with and not only the 6 ones plotted above.

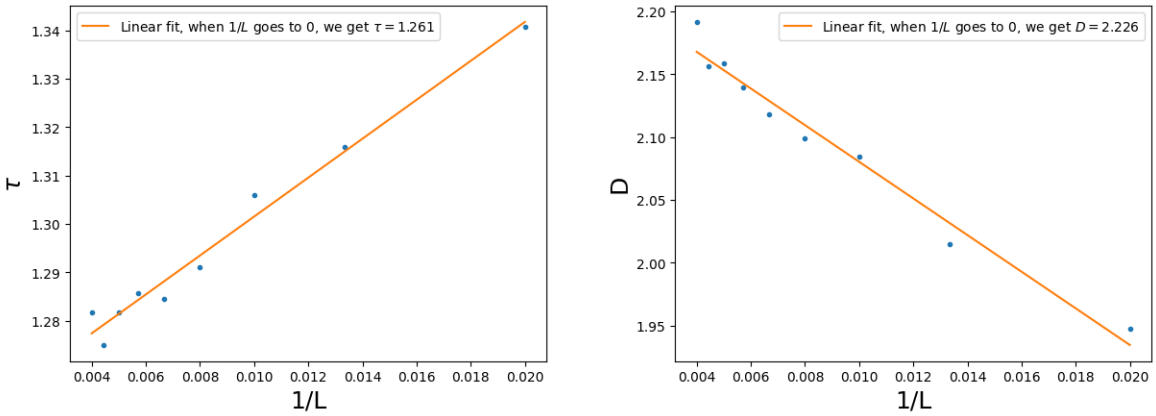

Figure 8: Plot of τ and D as functions of $1/L$.

Therefore, with this method we get as a conclusion a value of $\boxed{\tau = 1.261}$ and $\boxed{D = 2.226}$.

2. **Fitting the whole distribution**: For this other method, instead of fitting separately the linear and exponential parts, I fit at once the function

$$\ln(P(s)) = -\tau \ln s - \frac{1}{L^D} e^{\ln s}.$$

Again, I use scipy's curve-fit function for this and the results are shown in the following figures:

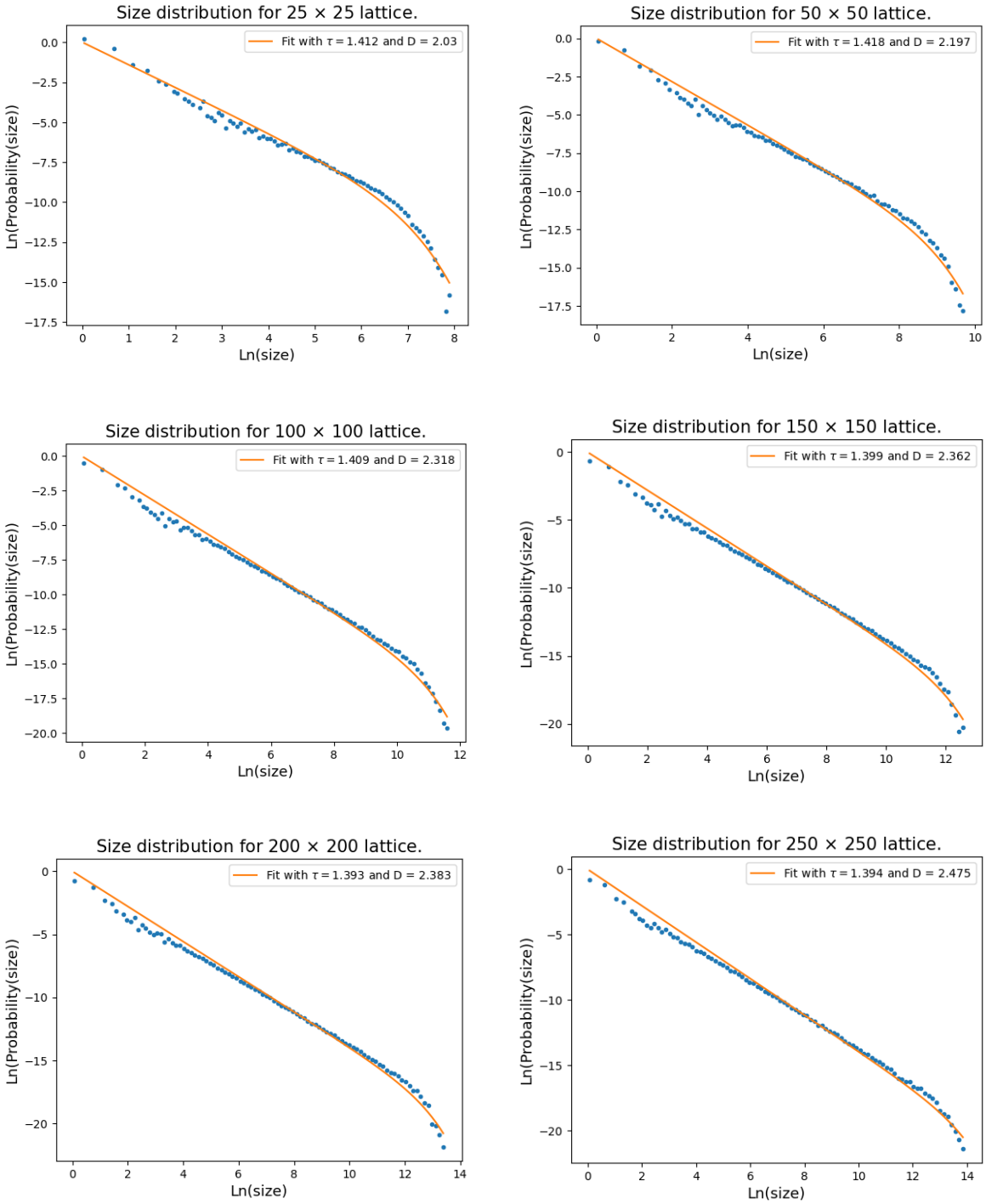

Figure 9: Size distribution plots for many lattices size along with the fit and the values of τ and D obtained with the fits.

Again, we can plot τ and D as functions of $1/L$ and see the values they tend to when $1/L \to 0$.

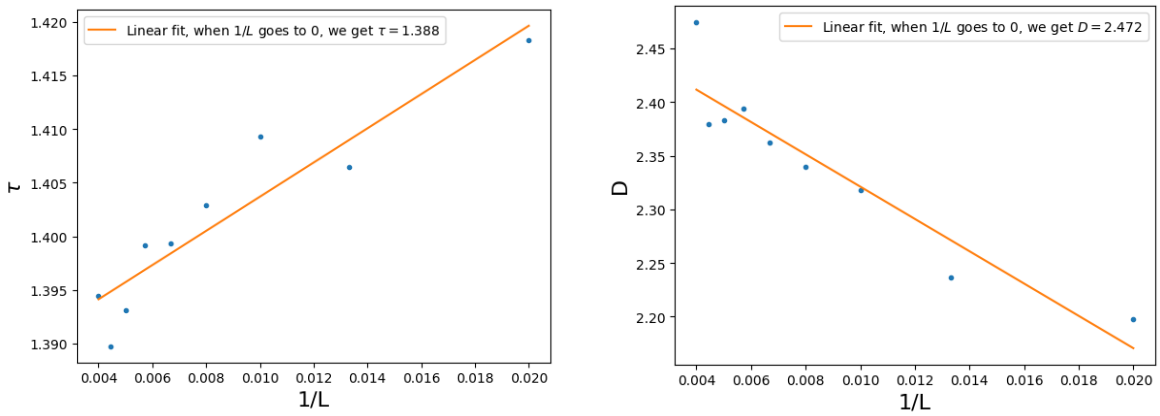

Figure 10: Plot of τ and D as functions of $1/L$.

Therefore, with this method we get as a conclusion a value of $\boxed{\tau = 1.388}$ and $\boxed{D = 2.472}$.

Therefore, with the two methods for fitting, we get slightly different values of τ and D. In both cases τ is around 1.3 and D around 2.3.

Part 3.

For a big lattice, then plot the size of avalanches (in number of relaxation events) as a function of the largest linear dimension of each avalanche (longest extension of avalanche along either the x-axis or y-axis). Deduce the dimension of the avalanche.

For this part I used a 250×250 lattice and after 60000 relaxation iterations, I did another 150000 iterations in which I calculated the size and length of the avalanches. To calculate the length of an avalanche, I just kept track of the leftmost, rightmost, topmost and bottom-most points in the avalanche and then took the biggest linear dimension.

According to what we know about fractals, the size of a fractal S will grow with its linear dimension L as

$$S \propto L^D$$

Therefore, plotting $\ln S$ vs $\ln L$, we should get a linear relation with slope D. The plot is shown in the following figure.

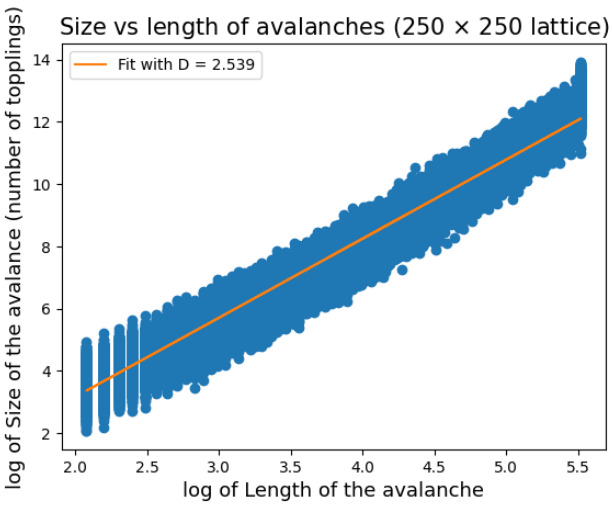

Figure 11: **Size of avalanches as a function of their linear dimension (in a log-log plot). We can see that the plot follows pretty clearly a linear relation and the best fit has a slope of $D = 2.539$.**

Fitting the plot to a line gives a slope of $\boxed{D = 2.539}$. To make the plot and calculate the fitting line, I ignored small avalanches (sizes below e^2) since they didn't quite fit with into a line with the rest of the plot. This is actually reasonable, since we know that when we define fractal dimension this way, we need to consider the limit as the length goes to infinity.

The value of $D = 2.539$ is quite close to the values of D we got on the second part of the assignment by fitting the distribution of sizes. So that these two methods of obtaining D give similar results.

It makes sense that we get a dimension that is slightly above 2, since the avalanches typically spread all over the lattice, so that the dimension should be at least 2. Furthermore, since points can topple more than once, avalanches can be "thick", so we would expect a dimension above 2.

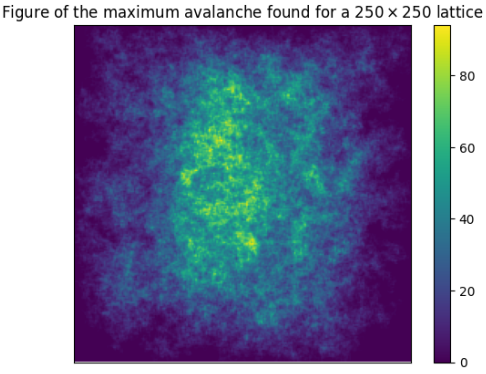

Figure 12: **Figure of the biggest avalanche I recorded for lattice 250×250. The color gives the number of toppling of each point.**

Part 4.

Simulate the system where grains are only added on the edge of the system and record the avalanche size distribution with this constraint. Deduce the corresponding exponent for these avalanches.

As instructed I did the same simulations as before but now only adding new grains to points in the border. First, I did a million repetitions for a 200×200 lattice to see the time it would take it to reach the steady state. Here is the graph of the sizes of avalanches and the cumulative sum of the sizes to identify the point at which we reach the steady state.

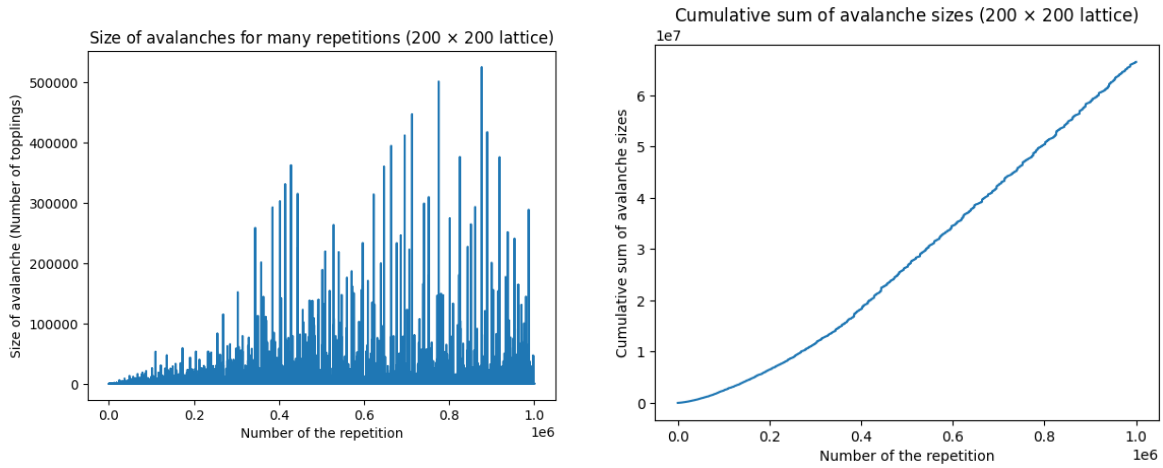

Figure 13: Sizes of avalanches for a 200×200 lattice. On the left side we can see a plot of the sizes themselves and on the right side the cumulative sum of said sizes.

From the figure, we can see that at around 400000 iterations, the system has avalanches that no longer grow in size and therefore the cumulative sum of avalanche sizes grows linearly.

To get the value of τ and D in this case, we can follow exactly the same procedure as before, that is, plot the logarithmic size distribution and fit it to the function $s^{-\tau}e^{-s/L^D}$. The results are shown in the following figure.

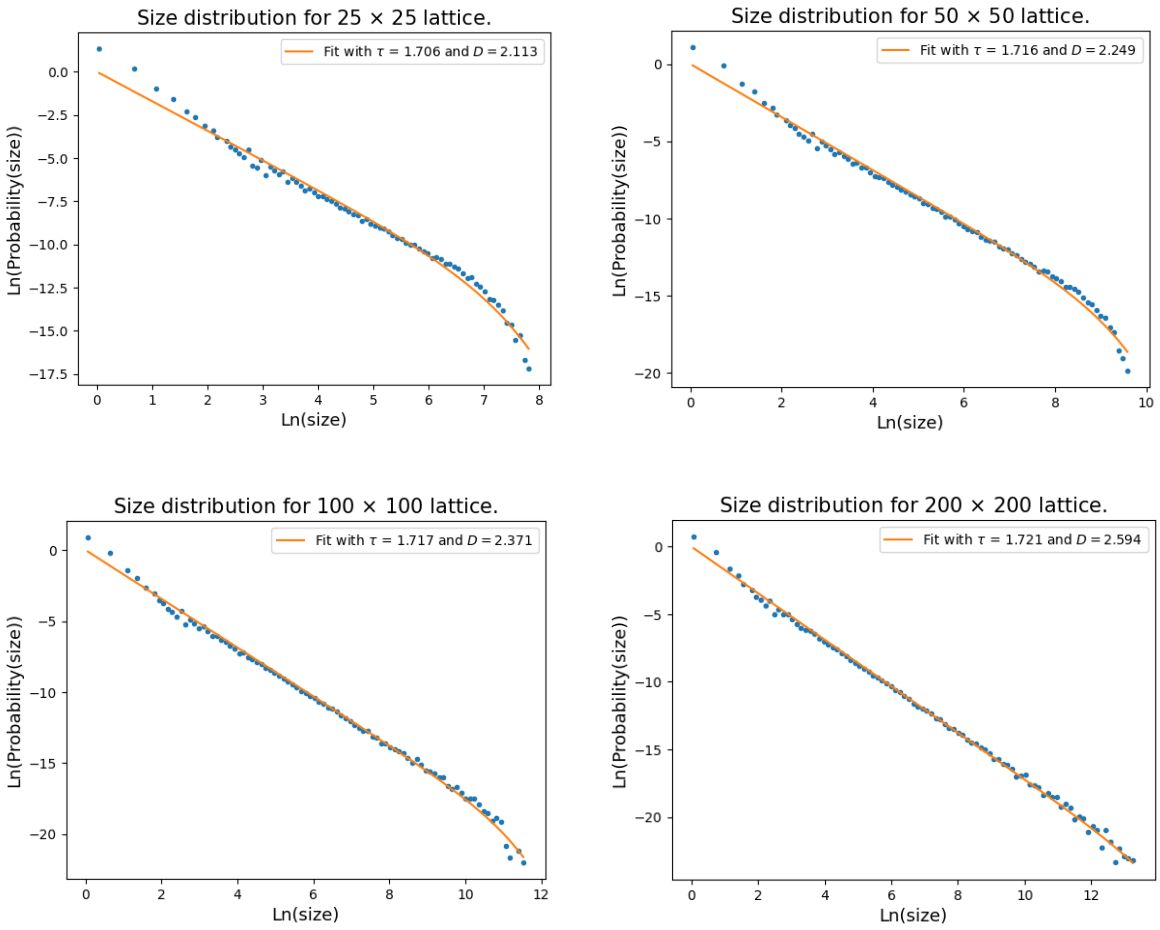

Figure 14: Size distribution plots for many lattice sizes along with the fit and the values of τ and D fitted.

In contrast with what we got for the case in which grains where added anywhere in the lattice, now the values of τ and D as functions of $1/L$ don't show a clear tendency and cannot really be approximated with a line as can be seen in the following figure.

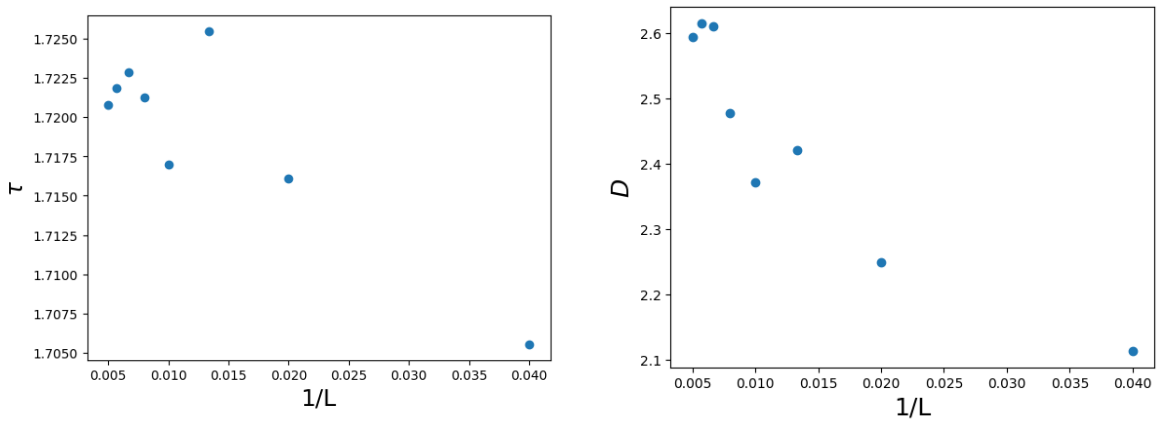

Figure 15: Plot of τ and D as functions of $1/L$.

Therefore, we will consider the values of τ and D to be those obtained for the largest lattice studied, which gives a result of $\boxed{\tau = 1.72}$ and $\boxed{D = 2.59}$.

Actually, we can also compute the value of D a different way just as we did in part three by searching for the exponent that describes the growth in size of avalanches as a function of their length. Doing this for a lattice of 200×200 gives as a result the following graph, from which we get a value of $D = 2.56$, very close to the value we have just found.

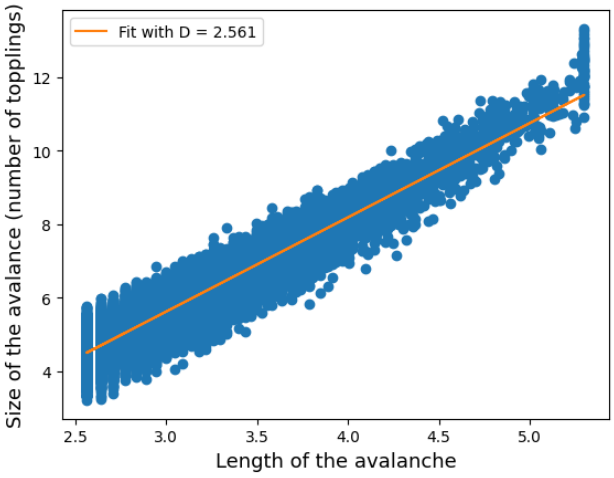

Figure 16: **Size of avalanches as a function of their linear dimension (in a log-log plot). We can see that the plot follows pretty clearly a linear relation and the best fit has a slope of $D = 2.56$.**

Part 5.

Additional points: Can you understand the scaling exponent of the boundary-driven avalanche in terms of the dimension of these avalanches?

To understand it, we can calculate the number of topplings of an avalanche in two different ways: The first one is by following the added grain and seeing it as a random walk and the second one is by using the size distribution $P(s) = s^{-\tau}e^{-s/L_D}$.

When we add the grain at the boundary, it follows a random walk and it will escape through the boundary when it first returns to its starting point. We know that the distribution of steps for a first return is $1/t^{3/2}$, so that the average amount of steps to return to the boundary is:

$$\int^{L^2} \frac{t}{t^{3/2}} dt = \int^{L_2} t^{-1/2} dt \propto t^{1/2} \Big|^{L^2} = L$$

The upper limit of the integral is L^2, since that is proportional to the number of topples in general that a grain has to participate in to get out of the system.

On the other hand, the number of topplings in an avalanche is on average

$$\int_0^\infty sP(s) = \int_0^\infty s^{1-\tau} e^{-s/L^d} \simeq \int_0^{L^D} s^{1-\tau} \propto s^{2-\tau} \Big|_0^{L^D} = L^{2D-D\tau}$$

Therefore, when equating both methods, we get that $L = L^{2D-D\tau}$ and therefore:

$$1 = 2D - D\tau \;\Rightarrow\; \tau = 2 - \frac{1}{D}.$$

We can see how closely our data follows that relation. We got the result that $D = 2.59$. According to the relation, we should have an exponent of $\tau = 2 - \dfrac{1}{2.59} = 2 - \dfrac{1}{2.59} = 1.61$. With the simulation, we got a value of 1.72, which is not so far off from 1.61.