Percolation of a 2D Lattice

Tahoe Schrader and Ksenia Sokolova
PHYS566

Abstract

In this assignment we computationally explore various $2D$ lattices and generate their spanning clusters, p_c and β. These variables refer to the probability a spanning cluster arises at and the critical exponent of a spanning cluster, respectively. Theoretically, these values for an infinite $2D$ lattice are $p_c \approx 0.593$ and $\beta = \frac{5}{36} = .138888$. Computationally, we were able to obtain $p_c = 0.594$ and a range $\beta = [.109436 \rightarrow 0.138344]$. These are reasonably close, which confirms that our code was successful.

Our code is available on GitHub:

`www.github.com/tahoeschrader/PHYS566_group4_projects`

1 Theory

Percolation theory describes the behavior of connected clusters. This theory has many important relations to physics and chemistry. For example, it can be used to describe the movement and filtering of fluids through porous materials. When a system *percolates*, a 2^{nd} order phase transition can also result.

The act of percolation refers to the existence of a spanning cluster. A spanning cluster is a cluster that reaches all edges of a physical boundary. For example, a spanning cluster must touch all four sides of a $2D$ box. Percolation, i.e. a system exhibiting a spanning cluster, is interesting because universality is *also* exhibited. Universality is when the properties of the system are independent of the systems dynamical details.

One feature of the percolating system that we would like to document is the clustering probability, p_c, where generally

$$p = \frac{\text{\# of occupied lattice sites}}{\text{total lattice sites}}. \tag{1}$$

Probability ranges and the theoretical results of such values are displayed in Table 1.

small p	isolated clusters
$p \approx 0.4$	many small clusters
$p \approx 0.6$	large, barely connected clusters
$p \approx 0.8$	most sites belong to same cluster

Table 1: The physical representation of various occupation probabilities on a lattice.

Another important value in percolation theory is the fraction of all sites in the spanning cluster with respect to all occupied sites

$$F = \frac{\#\text{ of spanning sites}}{\text{total }\#\text{ of cluster sites}}. \tag{2}$$

This value also takes the form

$$F = F_0 \left(p - p_c\right)^{\beta}, \tag{3}$$

where β is the critical exponent.

1.1 2D Lattice Systems

The steps for generating a $2D$ percolating lattice are quite simple:

1. Generate a 2D lattice
2. Populate a single lattice site at random and define it to be a cluster
3. Populate another single lattice site at random
   (a) If the new site touches an old site, add it to the cluster
   (b) If the new site doesn't touch an old site, define it to be a new cluster
   (c) If the new site touches multiple different clusters, choose the smaller numbered cluster to "win" and overtake the other clusters
4. Repeat until a spanning cluster arises

It is evident that larger values of p will more likely result in the existence of a spanning cluster than smaller values of p. Interestingly, the transmission from many clusters to a spanning cluster is sharp, qualifying it as a phase transition. For an infinite $2D$ lattice, the theoretical probability where this occurs is

$$p_c \approx 0.593 \tag{4}$$

whereas the critical exponent of the spanning cluster fraction is

$$\beta = \frac{5}{36}. \tag{5}$$

In Section 2, we computationally explore such a $2D$ lattice and generate spanning clusters, p_c and β for various lattice sizes.

2 Results

2.1 Extracting p_c

First we determine the critical probability by running the code for $N = 5, 10, 15, 20, 30, 50, 80$ for 50 times, every time finding the p_c.

To do this computationally, we used the cluster labeling method outlined above. Before running the averaging step, we tested the performance of our code by checking the dependencies between the time required to run one iteration and the size of the matrix. Please see results on Figure 1. Interestingly, these speeds are significantly faster than earlier version of our code. The time dependencies of an earlier version of our code can be seen in Figure 2. This speedup was achieved by removing nested for-loops in the spanning cluster check function, and by only generating new clusters where an old cluster did not already exist.

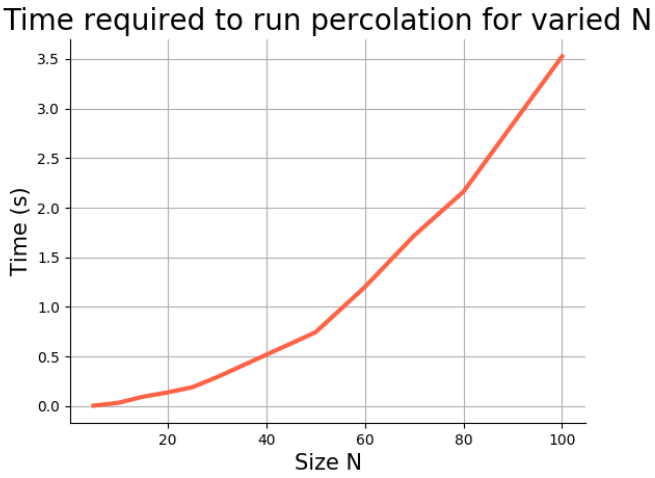

Figure 1: Time elapsed and size of the matrix

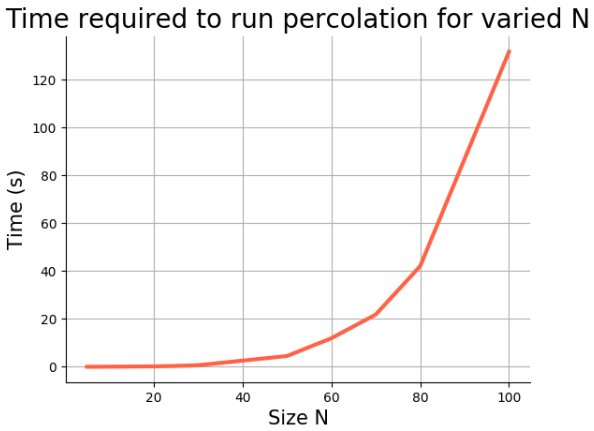

Figure 2: Time elapsed and size of the matrix for an earlier version of the code

We observe that our code is complexity $O[N^2]$, this is expected from the algorithm. This means that the larger the matrix, the worse the code performs, with squared scaling. And the change for the bad side happens relatively quickly. While this is durable for the cases we are observing, for larger matrices we would need to switch to a different algorithm.

Before the code calculates p_c, we can visually observe formation of clusters. (Note: on github project you can also find a .gif of the process). On the Figures 3 to 5 we see the development of the cluster for $N = 30$. In the beginning, as expected, we see many small clusters. As p increases we observe formation of larger clusters, but still no spanning cluster. And at the end we see one large spanning cluster, and just a couple of smaller clusters.

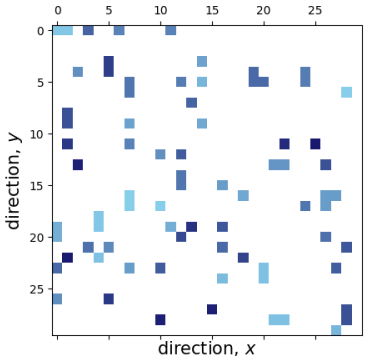

Figure 3: 2D percolation start

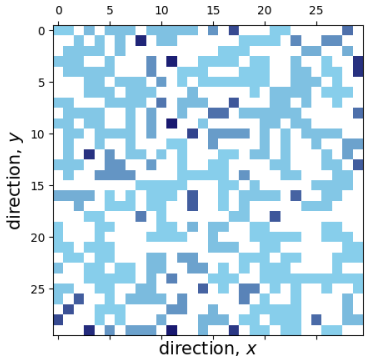

Figure 4: 2D percolation intermediate

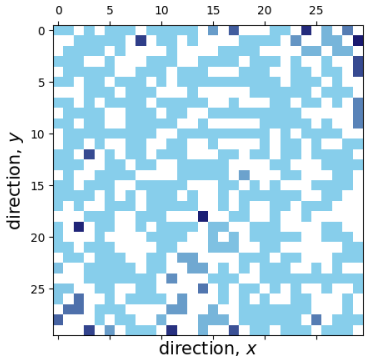

Figure 5: 2D percolation spanning cluster

To find critical probability, we ran the code for $N = 5, 10, 15, 20, 30, 50, 80$ for 50 iterations, averaging for every N. The spanning clusters for these N values are given in Figure 6 - 12.

After extracting an average p_c for each N, we extrapolated to find the critical probability. Please see the resulting plot attached (Figure 13), which shows a critical probability of 0.594. This is very close to the theoretical value of 0.593.

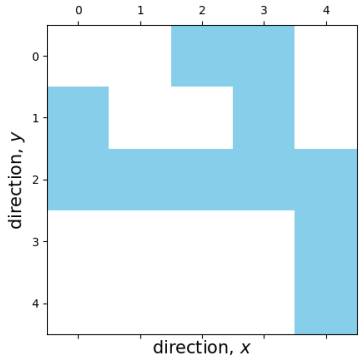

Figure 6: 2D percolation at $N = 5$

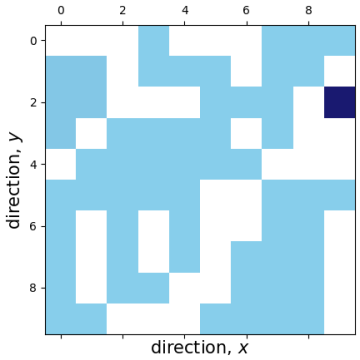

Figure 7: 2D percolation at $N = 10$

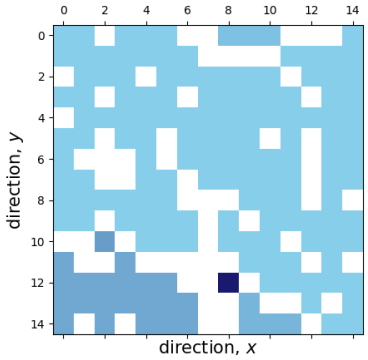

Figure 8: 2D percolation at $N = 15$

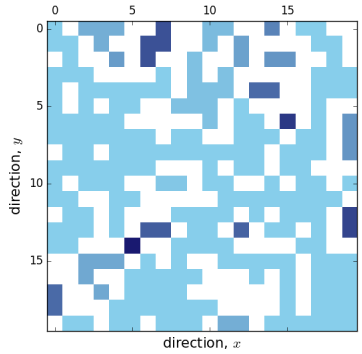

Figure 9: 2D percolation at $N = 20$

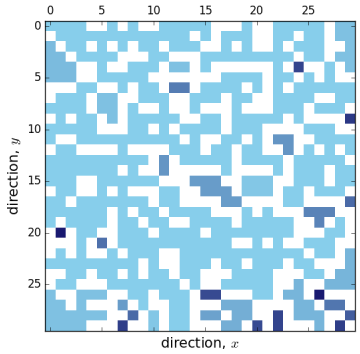

Figure 10: 2D percolation at $N = 30$

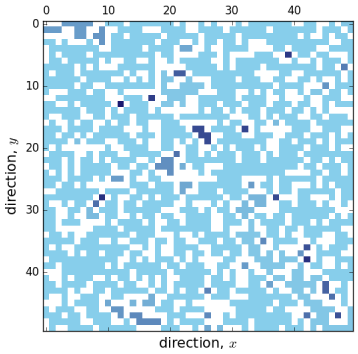

Figure 11: 2D percolation at $N = 50$

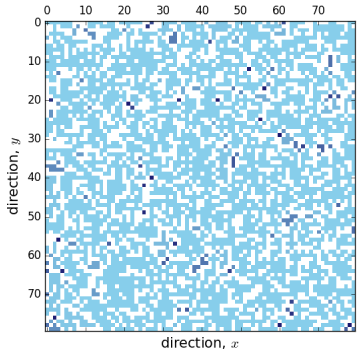

Figure 12: 2D percolation at $N = 80$

2.2 Extracting β

Next, we fixed $N = 100$ and computationally obtained parameter β. To do so, we implemented the base code also used in part A. First, we reach the matrix with the spanning cluster (this was the result of running part a). Then, we add points one by one to the matrix,

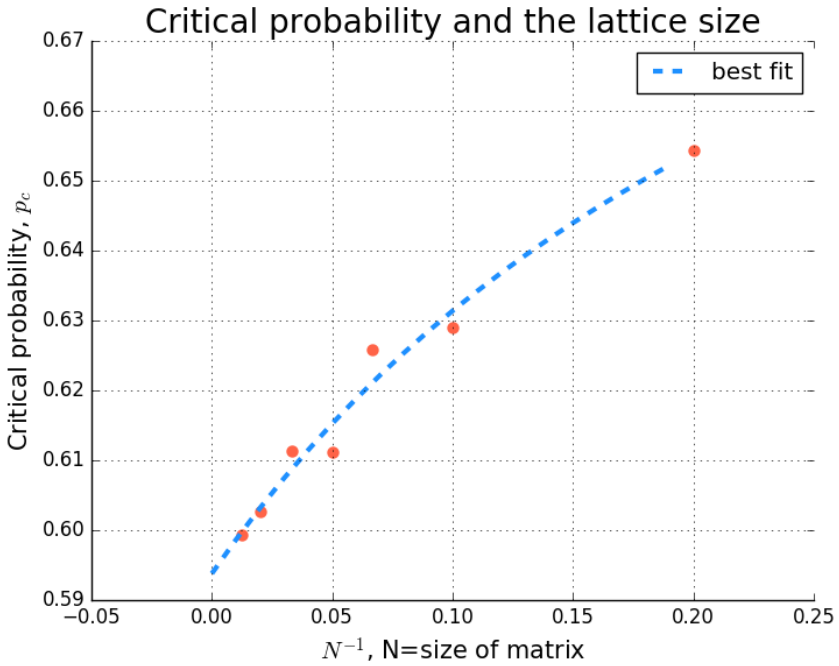

Figure 13: Critical probability plot for 2D percolation

and record values F and p. F is calculated according the the formula:

$$F = \frac{number\ sites\ in\ cluster}{total\ number\ occupied\ sites} \tag{6}$$

And p is just the percent of occupied sites, or the concentration, if we consider the meaning of this parameter.

To speed up the calculations, we actually used some trick, like counting the total number of elements added as a counter, and not parsing though the matrix every step.

To average over all p, we had to make sure the vectors from each run are all the same length. The discrepancy in the length of the vectors for F and p from every of the 50 runs can arise from the randomness in the cluster formation. The concentration at which the spanning cluster was reached varies through every run. To avoid this error, we truncate the averages based on the highest p_c achieved. We also average over all p_c to make sure we can use the most precise value for finding β.

After we have found the averaged p and F values, we are looking for a power law of the form:

$$F = F_0(p - p_c)^{\beta} \tag{7}$$

This is true when p is close to p_c, so we also take only first n values. We varied n to find the best amount of data to include. Experimentally, we found that leaving first 10% of the p and F values provides a good agreement between theoretical and experimental β.

To find a fit for the β we used a log transform:

$$log(F) = log(F_0) + \beta * log(p - p_c) \tag{8}$$

Please refer to the Figure 14 and 15 for the plots.

We ran the code for the $N = 100$ three times. The resulting β were

0.138344512943

0.10943672863

0.129361443917

Theoretical value of β in decimal format is 0.1388889

Therefore, our code consistently returned experimental β close to the theoretical values. The visual inspection of the plots also showed good fits.

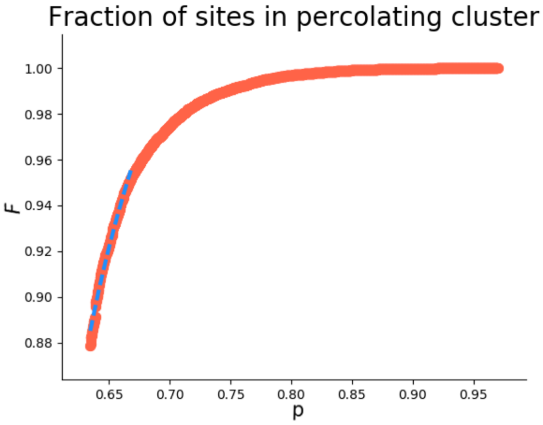

Figure 14: Fraction of sites in percolating cluster as function of concentration, N=100, average over 50 runs

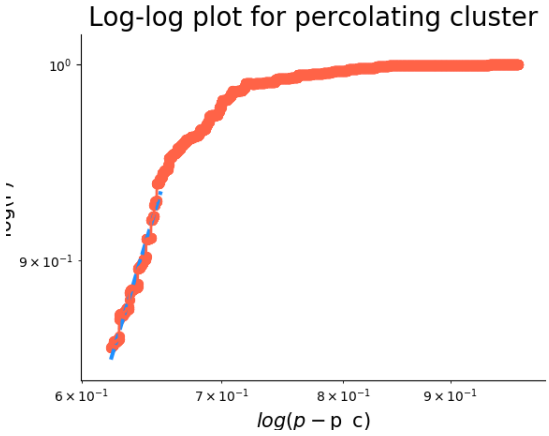

Figure 15: Fraction of sites in percolating cluster as function of concentration, N=100, average over 50 runs with log-log axes