

Lab 7 - Percolation

Thursday 12 October, 2006 - Due: Thursday 19 October

The aim of this lab is (1) to get you familiar with the computational implementation of a lattice and (2) to become familiar with some of the concepts in percolation. As for the latter, we will focus on the concept of clusters and the notion of a transition between a nonpercolating phase and a percolating phase. We will also measure the fractal dimension of the spanning cluster to compare it with the fractal dimension of the Sierpinski Triangle.

As always, please include in your writeup any sections of Python code you write.

1 A stroll through the code

Open `lab7.percolation.py` in IDLE and run it. You should see two different graphical outputs. One of the graphical outputs is an $x - y$ plot where x is the number occupied sites on the lattice and y is the size of the largest cluster. The second graphical output is like the one shown in class where the cluster configuration is displayed for one value of the occupation probability `p`. Each cluster is displayed in a different “color”.

How do we arrive at such outputs? Here’s an outline of the code. Lines 3-7 should be familiar to all of you. Lines 9-12 define several quantities: `L` is the length of the lattice, `N` is the maximum number of sites in the lattice (in two dimensions), `EMPTY` denotes that the site on the lattice is empty, or unoccupied, and `p`, again, denotes the occupation probability. Line 13 seeds the random number generator. Change the seed integer and you change the sequence of random numbers generated, otherwise, the sequence stays fixed for each run. You should already be familiar with lines 15-18.

Lines 21-27 define a recursive function that you will need later on to determine “who is connected to who”. That is all you need to know for today. Now, Line 30 initializes the two-dimensional integer array `nn`, which denotes the neighbors of a site on the square lattice. This array can be modified to construct other lattices. For now, the array sets up the square lattice with periodic boundary conditions. Lines 31-41 execute this set up.

Lines 44-52 set up a one-dimensional array, `order`. Initially, each site on the lattice is identified with one number (as opposed to two numbers in two dimensions). For example, the (0,0) site is identified 0, the (1,0) site is identified with 1, the (7,7) site is denoted as 63 when `L=8`. This identification is usually used for computational efficiency. The `for` loop in Lines 45-46 sets this identification. Then, the `order` array is randomized in Lines 47-51 to denote which site is occupied first, second, third, and so on. In other words, if `order[0]=29`, then site 29 on the lattice is occupied first, etc.

Now, Lines 53-75 determine the largest cluster size as the occupation of sites is increased one by one. This calculation involves the recursive function `findroot`. Line 77 plots the results of this calculation.

Lines 79-109, on the other hand, determines the cluster configuration for a fixed p , or fixed number of occupied sites. Hence, the `if` statement on Line 80, which sets up the randomized sites less than $p*N$ to be occupied by initializing the site in one-dimensional array `ptr` to `-1`. For those sites that are not occupied, the site in `ptr` is initialized to `EMPTY`. In other words, if site 8 (for $L=8$) is occupied, `ptr[8]=-1`. Remember site 8 is actually (0,1) in x and y coordinates (for $L=8$). Lines 79-109 are very similar to Lines 57-75. One, in principle, could combine the two. I have chosen not to here. However, you are welcome to optimize the code in ways that you see fit. Note that Lines 79-109 does not require the `nn` array. It sets up of the neighbors on the lattice “on the fly” so to speak, which differs from the earlier version where the array, `nn`, is needed.

Lines 111-128 plot the cluster configuration for a particular p . Each cluster is in a different shade of grey. You are encouraged to play with the color scheme to make it more to your liking. The `ptr` array contains the cluster configuration information in the following way: (1) If `ptr[i]=EMPTY`, then the site i is not occupied. (2) If `ptr[i]` is greater than or equal to zero, it identifies the cluster to which the site belongs to, and (3) If $0 > \text{ptr}[i] > \text{EMPTY}$, it identifies the negative of the size of the cluster. These sites are the roots of the each cluster. So, for example, if `ptr[3]=2`, then site 2 is the root site to which site 3 belongs to and `ptr[2]` should be some negative number (greater than `EMPTY`) that tells you the size of cluster labelled 2.

WHEW! Ok, if you got through all of that, then the rest should be “easy”.

2 Mucking around with the code

Please try the following:

- (Q1)★ Try several different values of p to see how the cluster configuration changes. Presumably, as p increases the disjoint clusters start to join to form larger clusters. This is also indicated by the x - y plot, but let’s look at the actual cluster configurations to check our intuition. At what p does there exist a *spanning cluster*? Where is this p on the x - y plot? Also, vary the seed for the random number generator and see if there are any changes in the critical occupation probability, i.e. the p at which the spanning cluster emerges.
- (Q2)★ Now increase L (double it, say) and compare the x - y plot with the smaller L . You may want to rescale the x - y plot by dividing both axes by N . By looking at the cluster configuration for the larger size, at what p does the spanning cluster emerge? How does this p compare with the smaller system size?
- (Q3)★ Now take a more careful look at the code and see how the square lattice with periodic boundary conditions is constructed. Write down an example of the square lattice on a piece of paper and write down which site corresponds to which i . Print out `nn` for the same length lattice that you have drawn and check that the print out matches

your expectation.

- (Q4)★ How can the code be modified to construct a triangular lattice? Remember the triangular lattice has coordination number 6 (as opposed to 4 for the square lattice). How can periodic boundary conditions be implemented? Modify the code accordingly and save it under a different name.
- (Q5)★ After successfully implementing the triangular lattice with your new code, change p and L to approximately determine where the critical occupation probability for each L . Is it different from the square lattice case? What about the shape of the x-y plot in the two cases (the square and the triangular)?

3 Fractal dimension of the largest cluster

We can measure the fractal dimension of the largest cluster (which is presumably the spanning cluster as soon as one exists (p is large enough)). To do this, fix p to be near the critical occupation probability. For different system lengths, determine the size of the largest cluster. Now make a log-log plot of the size as a function of L . The slope of the line gives you the fractal dimension. You can repeat each size measurement a number of times and use the average size of the largest cluster if your “line” is not smooth.

- (Q6)★ Compute the fractal dimension for the largest cluster on both the square and the triangular lattice.
- (Q7)★ Implement a third kind of lattice with periodic boundary conditions and repeat Q1-Q2. Also, compute the fractal dimension of the largest cluster. You can ask me for examples if you cannot come up with any. Also, try a google search and see what comes up.

Note: If you are having “problems” with the assignment do not hesitate to e-mail me at jschwarz@physics.syr.edu or call me at 607-342-0876.