UMEÅ UNIVERSITY
Department of Physics
Peter Olsson

May 19, 2021
Monte Carlo . . . in Physics, 7.5hp

Some simple stochastic models

Introduction

This computer lab has four different parts, 1 through 4. You will get one of these assigned by the teacher; for higher grade you should complete another one too, to you own liking. An extra exercise gives four bonus points.

1 Percolation

The task is to write a program that calculates the percolation probability as a function of the occupancy, $P(p)$. Consider a system with periodic boundary conditions. Generate random systems where the probability for each position to be occupied is given by p. (The number of occupied sites will thus fluctuate and be different for different configurations). Make a program that works in both two and three dimensions where the dimensionality is e.g. set with `#define D 2`. Use the following method with a fifo queue to check for percolation:

1. Start from each of the occupied positions at $y = 0$.
2. Put the position into the fifo queue.
3. Get a position (x, y) from the fifo.
4. For each nearest neighbor:

```
if (already visited) {
  if (that was with a different y) {
    percolation found
    exit
  }
}
else {
  mark the position as visited
  save the y coordinate
  put the position into the fifo queue
}
```

   Go to point 3.

To make this work one has to keep track of the y coordinate with a variable which is not modulo L. Percolation is found when the same position has been reached with two different values of y. (The difference should be L). If all the occupied points at $y = 0$ are tried as starting points without finding percolation one concludes that the system does not percolate.

For the report:

- Determine $P_L(p)$ for $L = 32$, 64, 128, and 256 for $0.5 < p < 0.7$ for $L = 32$ and more narrow intervals for the bigger sizes. Plot first the raw data $P_L(p)$ versus p. Second, plot versus $(p - p_c)L^{1/\nu}$ where $\nu = 4/3$ is known exactly. Adjust p_c (which is not known exactly, but $p_c \approx 0.593$) to get the best possible collapse of the data.

- Do the same in three dimensions but then with sizes $L = 12$, 16, 24, 32, and 48. First identify p_c from the crossing of the data and then adjust ν to get a decent data collapse in the plot of $P_L(p)$ vs. $(p - p_c)L^{1/\nu}$. (It is of course most fun to do this without first checking up values for p_c and ν from the internet.)

2 Cluster sizes in percolation

The idea is here to first identify clusters in site percolation and then used these clusters to determine the correlation length in a few different ways. Note that you are not expected to check for percolation in this exercise.

Consider the following hints for identifying clusters:

- We will restrict the study to $p < p_c$ and like to avoid the complications involved when identifying the spanning cluster. You should therefore make use of a very big system such that the possibility of a spanning cluster is vanishingly small. With $L = 1024$ and $p \leq 0.58$ it seems that one should never find a spanning cluster. Even though the system is big, your calculations should make use of results from a (large) number of such systems.
- Identify the clusters one by one and do the measurements described below. There is no need to save information related to several clusters.
- To identify clusters:
  - Scan through the system and let each occupied site be a starting point. (Make a site "unoccupied" when it has been identified as belonging to a cluster.)
  - Use some kind of queue to search through the system, much as you did in the Wolff Cluster update method.
  - To calculate the properties of the clusters it is convenient to use x and y variables that are not restricted to the range $[0, L-1]$.
  - Note that if $L = 2^n$, where n is an integer, one can get an image of x that is inside the simulation cell, i.e. $0 \leq x_{\mathrm{image}} < L$, by `x & (L - 1)`, where `&` is the bit-and operator.
- During each mapping of a cluster it may be good to keep track of $x_{\min}$ and $x_{\max}$, and similarly in the y direction, as a test that the system cannot percolate, without really testing for percolation.

Use three different methods to determine the correlation length:

1. First determine the correlation function $g(x)$ from the definition

$g(\mathbf{r})$ = the probability to find a connected path from an arbitrary occupied position $\mathbf{r}'$ to $\mathbf{r}' + \mathbf{r}$.

Note that to get random occupied positions you cannot take one such point per cluster; these random starting points have to be chosen independent of the identification of clusters.

2. Also determine ξ from
$$\xi_g^2 = \frac{\sum_c s_c^2 R_c^2}{\sum_c s_c^2},$$
where R_c^2 is the radius of gyration of cluster c and s_c is the number of sites that belong to the cluster.

3. Finally, use the method of Sec. 7.5, i.e. determine
$$\begin{aligned} g(0) &= \frac{1}{N} \left\langle \left| \sum_x n_x \right|^2 \right\rangle , \\ g(k_{\min}) &= \frac{1}{N} \left\langle \left| \sum_x n_x e^{-ik_{\min} x} \right|^2 \right\rangle , \end{aligned}$$
where $n_{(x,y)} = 0, 1$ for empty or occupied site, $n_x = \sum_y n_{(x,y)}$, and $k_{\min} = 2\pi/L$. The correlation length is then found from
$$\xi = \frac{1}{2\sin(\pi/L)} \sqrt{\frac{g(0)}{g(k_{\min})} - 1}.$$

3 Self-avoiding random walk

The task is to study self-avoiding random walk in two dimensions with three different methods:

1. First a determination of the exact value of $\langle S_N^2 \rangle$ for $N = 1$ through 4 through enumeration of all possible walks. Do this with pen and paper.

2. Second, a simple random generation which is aborted when the walk becomes self-intersecting. Try to get results with high precision for N up to 20. Compare with the first method to show that the program works correctly. The point with this part is just to get a way to check that the more efficient program in the next point produces correct results.

3. Use survival biasing to get values for N up to $N = 200$. Note that you will have to make a large number of runs to get good statistics for the largest N.

For the report:

- Make a table with the results, including error estimates, for $N = 1$ through 20 obtained with the three different methods (well, up to $N = 4$ for the complete enumeration).

- Plot $\langle S_N^2 \rangle$ versus N for the data from survival biasing on a log-log scale. We define the exponent ν by $\langle S_N^2 \rangle \sim N^{2\nu}$. Determine ν by fitting to

  $$\ln \left\langle S_N^2 \right\rangle = \text{const} + 2\nu \ln N.$$

  Do the fits for a few overlapping intervals: $N = 40 \ldots 80$, $N = 60 \ldots 100$, $\ldots$ up to $N = 160 \ldots 200$. Plot ν vs. N_{mid}, where N_{mid} is the midpoint for the N-values used in the fits. What is you final estimate of ν?

4 Complex networks

We here examine two different network models.

4.1 The Barabási–Albert model

This is a model of scale-free networks. The control parameters is the number of nodes N and the average degree m. We will here take $m = 4$.

1. Start with a graph of $m + 1$ nodes all connected to one another.
2. *Preferential attachment:* Add a node and m links attached to it. The other ends of the links should be attached to existing nodes i with a probability proportional to the degree k_i of i.
3. If the network has less than N nodes, go to step 2.

To implement the preferential attachment one can store all the links in an array, and: 1) Pick a link by random. 2) Follow it in a random direction to one of its nodes. Connect to that node. This is a means to get preferential attachment. (Why does it work?)

For the report:

1. Show that the time dependence of the degree of a node is, on average, $\sim t^{1/2}$, where t is the number of iterations after its creation.
2. Show, on a log-log plot, that the degree distribution is proportional to k^{-3}. The best way to do that (why?) is to plot the cumulative degree-distribution—the probability to find a node with degree larger than or equal to K as a functuion of K. This distribution should be proportional to K^{-2}.

4.2 The Watts–Strogatz model

This model of N nodes and $M = Nk$ links is defined as follows:

1. Connect a vertex i to $i - k, \cdots, i - 1$ and $i + 1, \cdots, i + k$. (Plus and minus is modulo N.)
2. Go through all links and rewire each link with probability p. (Do not accept a link that connects back to the same node or to a node which is already connected.)

For the report: We will here use $k = 2$.

1. Take $N = 1000$. Plot the clustering coefficient C (see below) and the average distance d as functions of p. Show that there is a region where C is rather large and d is small.

2. Study the p-dependence of d for $N = 100$, 200, 500, 1000, and p in the range 0.01 through 0.1. Can we conclude anything for $p = 10^{-4}$ without performing any simulations for that value of p? Is the large-N limit of $d(N, p)$ given by $\lim_{N\to\infty} d(N, p) = \text{const}$ or $\lim_{N\to\infty} d(N, p)/N = \text{const}$?

Technicalities: To measure the average distance d, go through all nodes one by one and measure the distance to the others by a breadth-first search. To calculate the clustering coefficent, use the following algorithm to measure c_{triangle} and c_{triple}:

1. Go through the nodes i.

2. Add $k_i(k_i - 1)/2$ to the count c_{triple} of connected triples.

3. Loop over all pairs of neighbors of i, say j, j'. If there is a link between j and j' increment the triangle count c_{triangle}. (This will triple count the number of triangles).

The clustering coefficient is then $C = c_{\text{triangle}}/c_{\text{triple}}$.