Modelling and Simulation 2015

Practical Assignment 2: Percolation

The so-called percolation model can be interpreted in many different ways. The name originally relates to a liquid *percolating* through a porous medium. Percolation can also be understood as an extremely simple randomized growth model, which does not include particle diffusion (in contrast to DLA). At a given time step, available neighbor sites of existing cluster are marked as *occupied* with fixed probability p, while with probability $1 - p$ they are marked as *empty*. Only unmarked neighbor sites of the cluster may be occupied, thereafter. The occupation probability p is the only parameter of the model.

We consider percolation growth on a square lattice system of size (2N+1)·(2N+1). The initial cluster consists of a single particle in the center, i.e. at position (N,N). All other sites are initially unmarked. The iterative growth process is summarized in the following pseudocode:

1) Determine all unmarked neighbor sites of the existing cluster, i.e. all sites in the lattice that have not been marked yet, but have at least one occupied nearest neighbor.
2) For each of the sites determined in (1), draw independently a random number $z \in [0, 1)$ and mark the site as *occupied* if $z \leq p$,
   as *empty* else.

Sites that have been marked as *empty* cannot be revisited and occupied anymore.
The cluster growth stops when

(a) all neighbor sites of the cluster are already marked as *empty*

or

(b) the cluster has reached the border of the lattice, i.e. any site in row or column number 1 or 2N+1 is occupied

In case (a), the cluster is called *non-percolating* or *finite*, while clusters that reach the boundary are termed *percolating* or (not very precisely) *infinite* clusters.

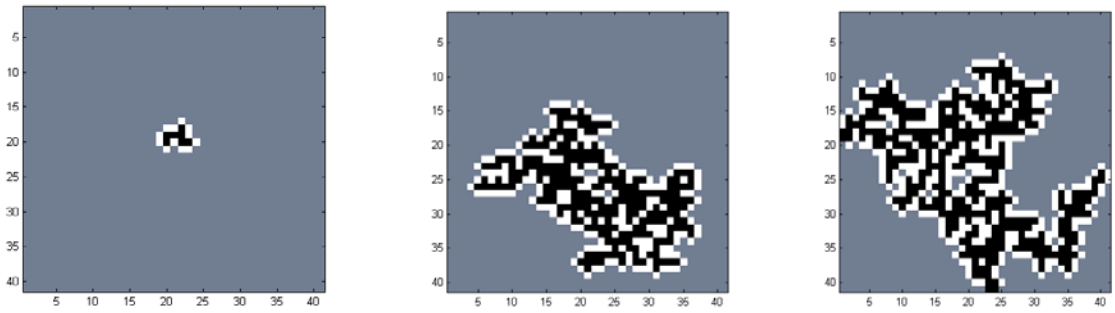

Figure 1: Percolation clusters on a square lattice of size 31 by 31. Occupied (empty) sites are marked in black (white), respectively. Sites shown in grey have not been visited in the growth process. Left: a small cluster, center: a larger but finite (non-percolating) cluster, right: a percolating (*infinite)* cluster that has reached the lattice boundary.

The above growth process can be implemented in a variety of ways. Obviously we can draw and store all random numbers z_{ij} beforehand (Matlab: `z=rand(2*N+1,2*N+1)`). It appears advantageous to work with lists containing the coordinates of all *occupied, empty*, and neighbor sites, but an explicit lattice representation is also possible. You should aim at the realization of a large number of growth processes for many different p in large lattices.

Core problems:

1) **Cluster size statistics**
   Perform percolation cluster growth on a lattice of 41 by 41 sites (or larger). For a number of values p in the range $0.3 \leq p \leq 0.7$ (in steps of 0.1 or smaller) perform at least 50 runs of the growth process. Of course you may also consider a wider range of *p*, but neither very small values nor $p \approx 1$ are particularly interesting, for obvious reasons.

   a) Determine the mean cluster size M and the corresponding standard deviation as a function of *p*. Important note: take only finite clusters into account, since the size of percolating clusters will depend on the boundary of the lattice and on the precise stopping criterion.

   b) Determine the fraction P_∞ of runs that yield an infinite cluster as a function of *p*.

2) **System size dependence**
   Repeat the experiments for *p=0.3* and *p=0.7* for (at least) two different lattice sizes, e.g $11 \cdot 11$ and $21 \cdot 21$. Observe and discuss qualitatively how the results change with the system size. If possible, speculate what the behavior could be in the limit of infinite lattice sizes.

A few remarks:

The recommended lattice sizes and number of runs are only (modest) estimates. The quality of the results will increase if you consider larger systems and if you perform statistics over more runs. It is obviously a good strategy to estimate the required runtime in test runs before starting the actual simulations. Please provide feedback if you run into problems with respect to computational power.

Percolation was first considered within the statistical physics community. It is an extremely simple example of a system displaying a so-called *phase transition,* i.e. a sudden change of macroscopic behavior with a control parameter, here it is *p.* In physics, temperature frequently plays the role of the control parameter, as for instance in the melting of a solid or the magnetization of iron. Percolation has served as an important model for the study of basic questions concerning phase transitions and continue to do so.

A popular interpretation of percolation is the *forest fire model:* a tree in a forest with mean density *p* (trees per square meter) is hit by lightning and the fire spreads to nearest neighbors. The question is whether the damage remains limited (non-percolating cluster of burnt trees) or essentially the entire forest burns (percolating). Of course you want to plant as many trees as possible (large p), but avoid large losses in case of a fire.

Possible bonus problems (these are only suggestions, own ideas welcome!):

Cluster identification: The original formulation of cluster statistics was slightly different: determine all connected clusters of sites with $z_{ij} \leq p$ in a square lattice; consider only clusters that do not touch the boundary. Use efficient cluster identification algorithms (*Hoshen Kopelman, Union/Find)* and convince yourself, that the observed statistics is essentially the same as in our consideration of single, central clusters.

Invasion Percolation: This particular attractive variant is parameter-free and constitutes a prototype example of *self-organized criticality.* It has been discussed in the context of avalanches or earthquakes. At every time step, we occupy only the unmarked neighbor site with the (currently) lowest random number z_{ij}. Thereafter, the list of neighbors is updated, and we proceed with the (now) lowest z_{ij}. Obviously, the process always yields a percolating cluster. We stop growth upon the first contact with the boundary and, then, determine a histogram of the z_{ij} corresponding to the cluster sites. Obtain the average histogram over many runs in large lattices and discuss its properties.

Fractal dimension: Determine the fractal dimension of (finite) clusters as a function of *p.* Employ the naïve estimation used for DLA or more sophisticated methods (e.g. *box-counting).*