Percolation in a Two-Dimensional Square Lattice

Daniel Bristow

May 5, 2021

1 Introduction

Percolation is a topic that can be used to describe various physical systems such as porous materials, forest fires, and phase transitions (such as sol-gel transitions).[1] What these problems have in common is that they can be represented by percolating lattices; that is, the lattices contain a number of growing clusters of occupied cells. After a certain amount of time (which for the purpose of the percolation problem is represented by the probability p that the cell of a lattice is occupied) there will be a cluster that eventually spans the entire lattice from opposing ends. This is known as a percolating cluster. For an infinite 2D lattice (of any type of cell, whether it be square, triangular, honeycomb, etc.), this cluster will form at around $p_c \approx 0.593$.[2]

2 Method

For simplicity, we are simulating an $L \times L$ square lattice. To simulate percolation, cells will be occupied one at a time. The probability p will denote the probability of a given cell of the lattice being occupied. The cells will be assigned one of three numbers.

- **0:** The cell is unoccupied (all cells are labeled 0 at the start).
- **1:** The cell is occupied but is not a part of the percolating cluster.
- **2:** The cell is unoccupied and is a part of the percolating cluster.

To identify the percolation cluster, Breadth First Search (BFS)[3] seeks a path of occupied cells (specifically cells labeled 1) starting from the most recently added cell that spans either both the left and right side of the lattice of the top and bottom of the lattice. Once the percolation cluster has been identified, BFS will only run again if a cell added is found to have a neighboring cell in the percolation cluster (a neighboring cell labeled 2; from here, BFS once again seeks out cells labeled 1 to merge with the percolation cluster).

3 Verification of program

Comparing to Figures 7.24, 7.25, and 7.27 in [2] shows similar results for various runs. Likewise, we can always be sure when watching the animation resulting from the program that the percolating cluster is identified precisely when a cell is added such that a cluster spans the lattice. This is seen when teal cells (labeled 1) begin to appear in place of previously yellow cells, since the yellow cells now identify the percolating cluster (they are now labeled 2; the blue cells are the remaining unoccupied cells labeled 0).

4 Data

We consider a run for a 300 × 300 lattice.

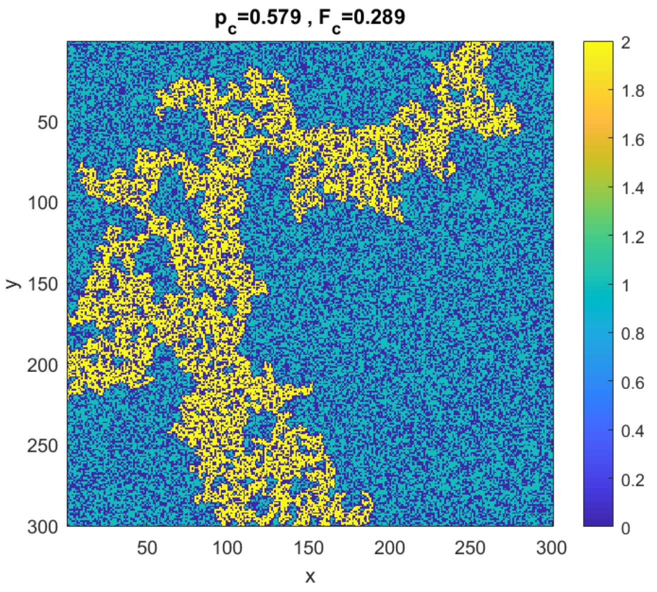

Figure 1: Lattice at the moment the percolating cluster is identified at $p_c \approx 0.60$, spanning from the top to the bottom of the lattice.

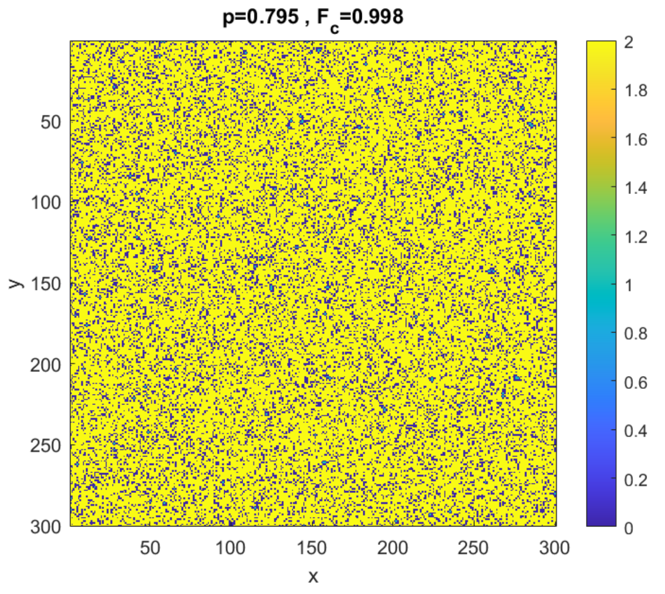

Figure 2: The lattice is just about nearly full at $p \approx 0.80$, with 99.8 percent of occupied cells being part of the percolating cluster.

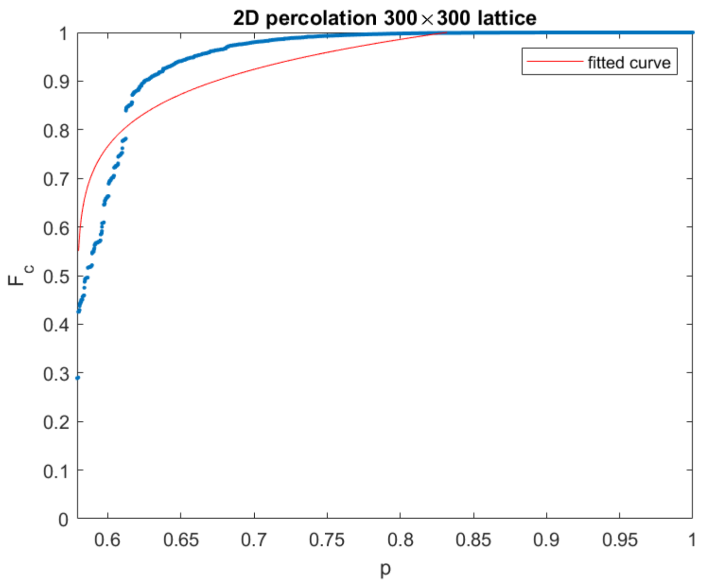

Figure 3: A non-linear fit of the fraction of occupied sites in the percolating cluster F_c versus probability of a given site being occupied p.

5 Analysis

For an infinitely large 2D lattice (of any type of cell), we find that $p_c \approx 0.593$. In the run of the program described here, $p_c = 0.57903$, which is in accordance with our expectations.

We are also interested in the fraction of the cells contained within the percolating cluster versus the total number of cells in the lattice. Once the percolating cluster is identified, we expect the relationship

$$F = F_0(p - p_c)^\beta \tag{1}$$

where for an infinitely large 2D lattice (again, consisting of any type of cell), we find $\beta = \frac{5}{36}$. F_0 is a proportionality constant and unknown alongside β. From the non-linear fit in Figure 3 based on equation (1), the program tells us that $F_0 = 0.159$ and that $\beta = 0.1071$, which again is in accordance with our expectations.

6 Critique

We note that the expectations we have described are for the case of an infinite lattice. Given the computational power required for this simulation, we have limited ourselves to a 300×300. Runs of this program with smaller lattice sizes have returned values for p_c as low as 0.45 and as high as 0.7, though the average is still around 0.6. Nearly all runs of the lattice at above $L = 100$ have returned p_c at around 0.6.

As for the value of β, runs of the 300×300 case often seem to be returning values quite a bit lower than the expected 0.1389 (with values for β as low as 0.05). This could again be due to the limits for the lattice size presented by the computational power demanded by the program, but the method for the non-linear fit we are using may very well be improperly implemented and should be investigated.

References

[1] Lucas Böttcher. *Introduction to Computational Physics.* Institute for Theoretical Physics, ETH, Zürich.

[2] Nicholas J. Giordano and Hisao Nakanishi. *Computational Physics.* Pearson, 2 edition, 2006.

[3] S. Dasgupta, H. Papadimitriou, and U. V. Vazirani. *Algorithms.* McGraw-Hill Education, 1 edition, 2006.