FYS4460 Project 3

Svenn-Arne Dragly

In this project we are working with percolation - the study of connectivity and other properties in random media. We start out with a matrix of uncorrelated randomly occupied sites.

I. INTRODUCTION

Percolation is the study of connectivity and other properties in random media. We will in this project study a two-dimensional system where a binary matrix is chosen at random with a given probability p for each of its sites to be occupied or not. These sites will be grouped into clusters which are studied for their connectivity, areas, masses and other properties. In this project, we will dig into topics such as percolation treshold and power law distributions.

II. THEORY

A. Definitions

For the reference (and to give myself somewhere to look up stuff), I've listed the definitions used throughout this project in table I.

Term	Definition
cluster	Set of sites that are connected
connected	Sites that share sides. In this project, we define connections with 4 neighbors on a square lattice
spanning cluster	A cluster that spans the whole system from edge to edge in either direction
finit cluster	Any cluster that is not a spanning cluster

TABLE I. Definitions used throughout this project

There are also a set of properties that we will explore, such as the probability for a site to be part of the spanning cluster and the probability for a spanning cluster to exist at all. These are listed in table II.

Property	Definition
p	Probability that a site will be occupied.
$P(p, L)$	Probability that a site is part of the spanning cluster
$\Pi(p, L)$	Probability that a spanning cluster exists in the system
$g(r, p, L)$	Probability that two sites a distance r apart belong to the same spanning cluster
s	Denotes the size of a cluster
$\xi(p, L)$	Correlation length
$sn(s, p, L)$	Probability for a site to belong to a cluster of size s.
$n(s, p, L)$	Cluster density - a useful property that among other things is used to find $sn(s, p, L)$
$S(p)$	Average cluster size.

TABLE II. Properties that will be explored in this project

B. One-dimensional percolation theory

We use a few findings from one-dimensional percolation theory to build up our intuition for finite-dimensional percolation theory.

1. Verifying the normalized density

From one-dimensional percolation theory, we need to verify that $sn(s,p)$ is a normalized density. This is discussed in section 2.2 of the syllabus, and will be elaborated here for future reference.

$$\sum_s sn(s,p) = \sum_{s=1}^{\infty} sp^s(1-p)^2 = (1-p)^2 p \sum_{s=1}^{\infty} sp^{s-1} \quad (1)$$

Adding a term for $s = 0$ to the sum changes nothing because $0 \cdot p^{0-1} = 0$. This lets us rewrite the above as

$$\sum_s sn(s,p) = (1-p)^2 p \sum_{s=0}^{\infty} sp^{s-1}. \quad (2)$$

This is written with the extra p outside the sum to exploit the derivative of p^s,

$$\sum_s sn(s,p) = (1-p)^2 p \frac{d}{dp} \sum_{s=0}^{\infty} p^s. \quad (3)$$

Further, we use the summation rule $\sum_{s=0}^{\infty} p^s = 1/(1-p)$ to find

$$\sum_s sn(s,p) = p. \quad (4)$$

This shows that $sn(s,p)$ is a normalized density.

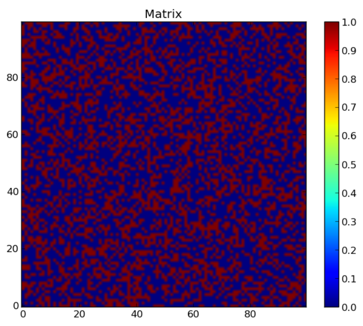

FIG. 1. Example of a binary matrix that we'll perform our calculations on.

III. IMPLEMENTATION

Python has been used as the main programming language for this project. To do this, I needed to find the relevant functions to use in Python that corresponds to the listed functions in MATLAB.

A. Working with percolation clusters in Python

For the original article, see this webpage.

First of all, we need to include the pylab package together with the scipy.ndimage.measurements package:

```
from pylab import *
from scipy.ndimage import measurements
```

Now, let's create a random matrix of filled and unfilled regions and visualize it:

```
L = 100
r = rand(L,L)
p = 0.4
z = r<p

imshow(z, origin='lower')
colorbar()
title("Matrix")
show()
```

This creates a matrix of random numbers r before creating a map z of this matrix where each element is set to True if $r < p$ and False if $r > p$ as shown in figure 1.

Furthermore we want to label each cluster. This is performed by the scipy.ndimage.measurements.label function:

```
lw, num = measurements.label(z)
imshow(lw, origin='lower')
colorbar()
title("Labeled clusters")
show()
```

This labels each cluster with a number which may be visualized using the imshow function as shown in figure 2.

Beacause the label starts from the bottom up, the cluster colors are a bit too ordered, making it hard to distinguish two clusters close to each other. To fix this, we may shuffle the labeling with the following code:

```
b = arange(lw.max() + 1)
shuffle(b)
shuffledLw = b[lw]
imshow(shuffledLw, origin='lower')
colorbar()
title("Labeled clusters")
show()
```

Now it is way easier to see the different clusters as shown in figure 3.

After this we may want to extract some properties of the clusters, such as the area. This is possible using the scipy.ndimage.measurements.sum function:

```
area = measurements.sum(z, lw,
     index=arange(lw.max() + 1))
areaImg = area[lw]
im3 = imshow(areaImg, origin='lower')
colorbar()
title("Clusters by area")
show()
```

The above code now plots the same matrix as above, but this time with all clusters colored by area.

Finally, we may want to find the bounding box of the largest cluster, so we again may see if there is a path from one side to the other. This is possible by using the function scipy.ndimage.measurements.find_objects. Note however that this is a bit risky. If there are two clusters that are largest with the same area, find_objects will find the bounding box of both clusters. That's why we'll need to rather loop over the labels for the clusters with max areas. This is done by the following code:

```
labelList = arange(lw.max + 1)
```

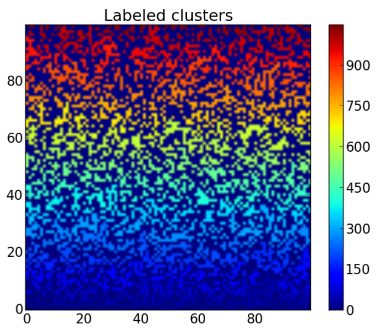

FIG. 2. Matrix labels, with colors ordered from bottom to top.

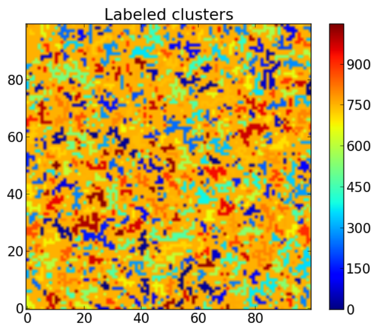

FIG. 3. Matrix with labels in random order.

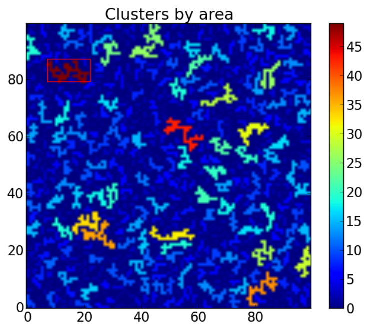

FIG. 4. Matrix with clusters colored by area.

```
maxLabels = labelList[where(area == area.max())]
for label in maxLabels:
  ...
```

In this for loop, we use the find_objects function to plot a rectangle around the largest clusters:

```
sliced = measurements.find_objects(lw == label)
if(len(sliced) > 0):
    sliceX = sliced[0][1]
    sliceY = sliced[0][0]
    plotxlim=im3.axes.get_xlim()
    plotylim=im3.axes.get_ylim()
    plot([sliceX.start, sliceX.start, sliceX.stop,
          sliceX.stop, sliceX.start], \
                     [sliceY.start, sliceY.stop,
                         sliceY.stop,
                         sliceY.start,
                         sliceY.start], \
                     color="red")
    xlim(plotxlim)
    ylim(plotylim)

show()
```

This final plot is shown in figure 4 and the whole script is located in the Git repository for this project.

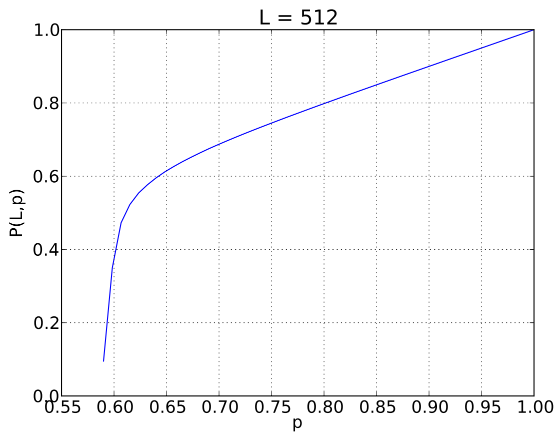

FIG. 5. Probability for a site to be part of the spanning cluster $P(p, L)$ plotted as a function of p. This was found by sampling P for 500 random binary matrices for different values of p.

B. Probability for a site to be part of the spanning cluster

To find the probability for a site to be part of the spanning cluster, $P(p, L)$, I've iterated over different p in the range $p = 0.5$ to $p = 1$ (by inspection, there was no reason to include values for $p < 0.5$). For each p, the above program is run 1000 times while the **sliceX** and **sliceY** variables are checked against the size of the whole matrix:

```
if sliceX.stop − sliceX.start >= Lx or sliceY.stop
     − sliceY.start >= Ly:
    P = area.max() / (Lx * Ly)
else:
    P = 0
```

We then find the average of all P for each p. In principle this could easily be extended to systems of rectangular shape. To extend it further to systems with for instance hexagonal boundaries or shearing, further steps would be needed to compare the edges of the boundaries to the spanning cluster. This is likely just a matter of some geometric operations, for instance creating bounding boxes near the boundaries and seeing if these intersect the edge or not. This has not been attempted in this project.

For the case with $L = 100$, P goes as a function of p as shown in figure 5.

C. The form of the function

We now have an estimate for the probability for a site to be part of the spanning cluster for different values of p. From here we may want to see how well this fits the theory which says that for $p > p_c$, the probability $P(p, L)$ for a given site to belong to the percolation cluster will

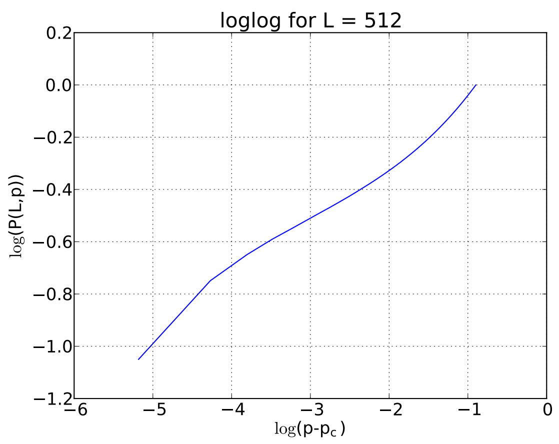

FIG. 6. loglog plot of the probability for a site to be part of the spanning cluster $\log P(p, L)$ plotted as a function of $\log(p - p_c)$. This was found by sampling P for 500 random binary matrices for different values of p.

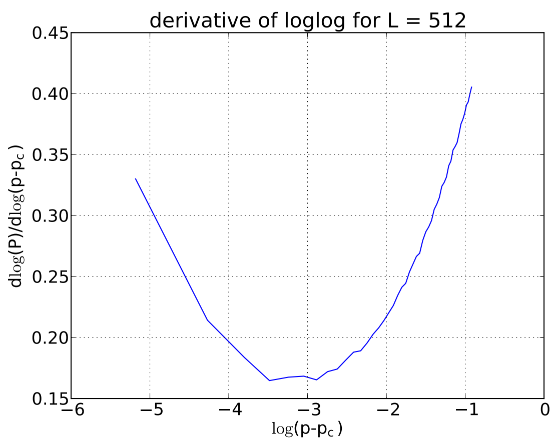

FIG. 7. Derivative of the loglog plot of the probability for a site to be part of the spanning cluster $\log P(p, L)$ plotted as a function of $\log(p - p_c)$. This was found by sampling P for 500 random binary matrices for different values of p.

be on the form

$$P(p, L) \sim (p - p_c)^{\beta}. \quad (5)$$

To put this to the test, we take the logarithm on both sides to find

$$\ln P(p, L) \sim \beta(p - p_c). \quad (6)$$

Meaning that we should be able to read off the gradient in a loglog plot of $\log P(p, L)$ vs $\log(p - p_c)$. Such a plot is shown in figure 6. By taking the derivative of this function (numerically) we get the plot shown in figure 7. From this plot we may read off the lowest value of the derivative as an estimate for β. In this case it appears to be about

$$\beta = 0.17. \quad (7)$$

This is a bit higher than what is the expected value of $\beta = 0.14$.

D. Power-law distributions

Before addressing cluster number densities, we need some tools to help us determine the exponent of power-law distributions. To do this, we are creating a set of random data points in Python:

```
z = rand(1e6)**(-3+1)
```

This distribution is not suited for a regular histogram, but one with logarithmic binning will do. We may therefore construct a series of bin edges determined by Python's **logscale** function:

```
mybins = histogram(z,
     bins=logspace(0,log10(z.max()),100))
```

We may now calculate the cumulative distribution function with data points within this binning by using Python's **cumsum** function and divide by the sum of all bins to find $P(Z > z)$:

```
cumulativeDistribution = cumsum(mybins[0]) /
     float(sum(mybins[0]))
```

The result is shown in figure 8. Taking the derivative of the cumulative distribution gives us the actual distribution

$$f_Z(z) = \frac{dP(Z > z)}{dz} \quad (8)$$

After calculating this, we take the logarithm of both $f_Z(z)$ and z and calculate the incline of the curve. To find this exactly, we may take the numeric derivative and plot it as shown in figure 9. Here we clearly see that the value for α is approximately $\alpha = -0.5$.

With these tools in place, we can move on to calculating the cluster number density.

E. Cluster number density

The cluster number density $n(s, p)$ can be calculated by running the program we used in the first exercises to find the area for each cluster. Thereafter, we may find the number of occurences each area and weight each occurence by the area itself using the NumPy histogram function:

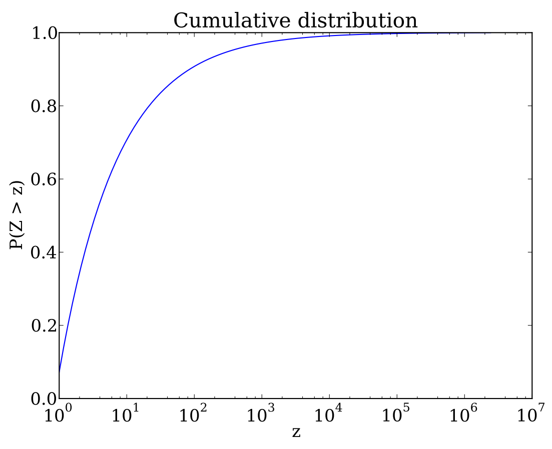

FIG. 8. Cumulative distribution of the randomly selected data points.

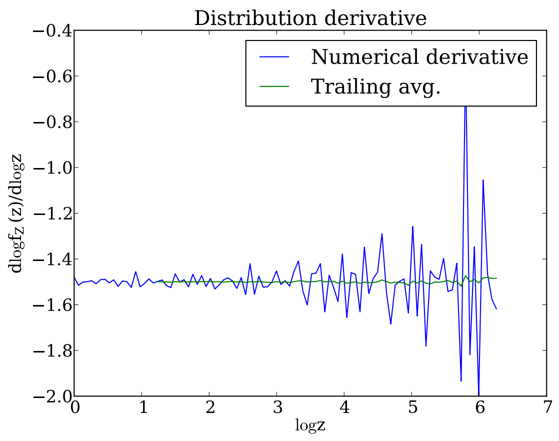

FIG. 9. Derivative of the logarithm of the cumulative distribution of the randomly selected data points.

```
currentBins = histogram(area, bins=bins,
     weights=area)[0]
```

Before doing this, we set the area of the spanning cluster to zero, just to make sure it does not contribute to the statistics.

If we now divide by the total area of the system, we will find the cluster number density for this specific system.

It is interesting to see what happens to the cluster number density as p approaches p_c from above and below. This is shown in figure 10.

F. Finding cluster number density scaling

Furthermore, we are interested in seeing how the cluster number density at p_c varies as a function of s with the system size L. This is shown in figure 11. This is interesting because we assume that $F(s/s_\xi) = 1$ at p_c. This is what we find in 1-dimensional percolation theory, where $F(s/s_\xi) = e^{-s/s_\xi}$ and $s_\xi = -1/\ln p$. When $p = p_c$, we get $s_\xi = \infty$ and assume something similar to happen in 2-dimensional systems.

With $s/s_\xi = 0$, we get $F(s/s_\xi) = 1$ and the cluster number density at p_c becomes only

$$n(s, p_c) = s^{-\tau} \tag{9}$$

In all, we may use this to find τ. In figure 11 we see that with increasing L, there is a loglog trend towards a line with decreasing incline. If we could increase L to extreme sizes, we may estimate the true value of τ, but we are pretty much stuck with $L \leq 2048$ for this project. However, we could plot the estimate τ as a function of L and see if this converges to the real value.

This is done in figure 12. We see that τ goes toward the true value of $\tau = 187/91 \approx 2.055$, but it would be hard to deduce this without knowing τ beforehand. In principle it is possible to use numerical methods to fit the curve in this figure, but it appeared somewhat tedious to do and was therefore not done in this project. The most clever method appears to be to use SciPy's `curve_fit` function. For now, the best we can guarantee from our data is that

$$\tau > 1.94. \tag{10}$$

G. Scaling of characteristic cluster size

If we assume that the characteristic cluster size is proportional to $p - p_c$ through

$$s_\xi \sim |p - p_c|^{1/\sigma} \tag{11}$$

we may plot s_ξ against $|p - p_c$ to find σ. To do this, we first have to gather some data on the relation between the two.

We start out with the definition

$$n(s, p) = s^{-\tau} F(s/s_\xi) \tag{12}$$

to find

$$\frac{n(s,p)}{n(s,p_c)} = \frac{s^{-\tau}}{s^{-\tau}} \frac{F(s/s_\xi)}{F(0)}, \tag{13}$$

where we have used that $s_\xi = \infty$ at $p = p_c$. Further we assume that $F(0) = 1$ in 2 dimensions as it does in 1 dimension. This allows us to rewrite the ratio as

$$\frac{n(s,p)}{n(s,p_c)} = F(s/s_\xi). \tag{14}$$

This is plotted in figure 13 as a function of s.

We don't yet know the function F, but we may use that we expect the function to go drastically towards 0 as $s \to s_\xi$. If we intersect F at a given value that will be

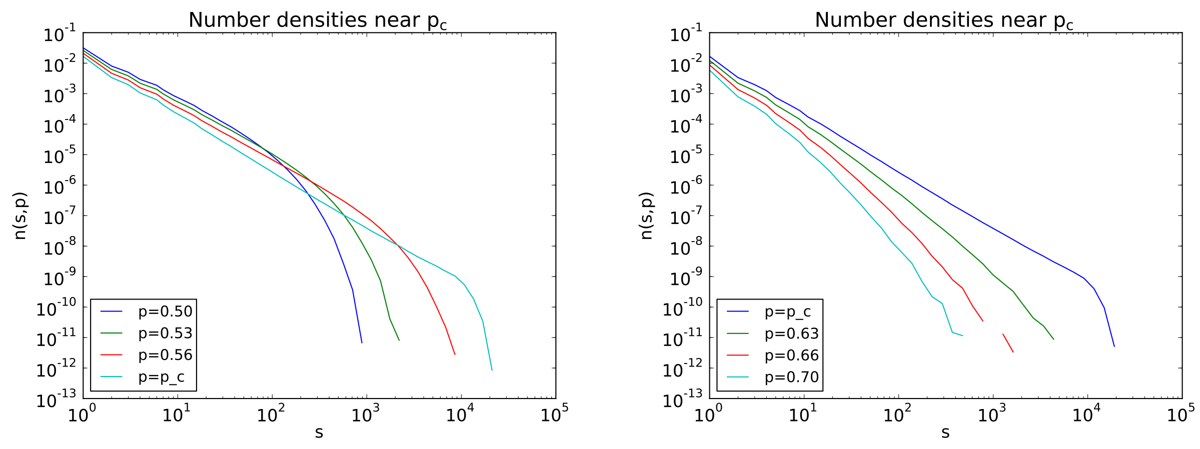

FIG. 10. loglog plot of the cluster number density for a system with $L = 256$ using 1000 samples for each value of p as we approach p_c from below (left) and above (right).

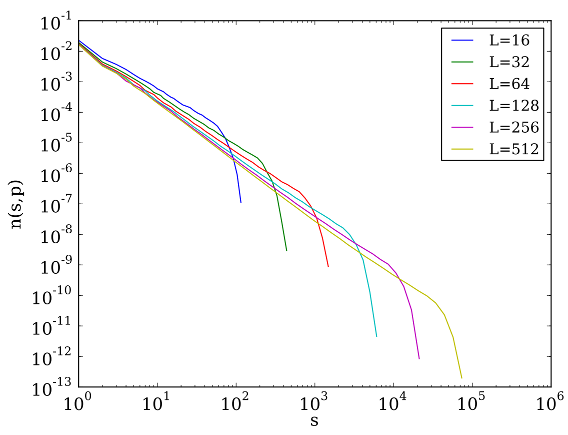

FIG. 11. Cluster number density $n(s, p_c)$ as function of s for different system sizes.

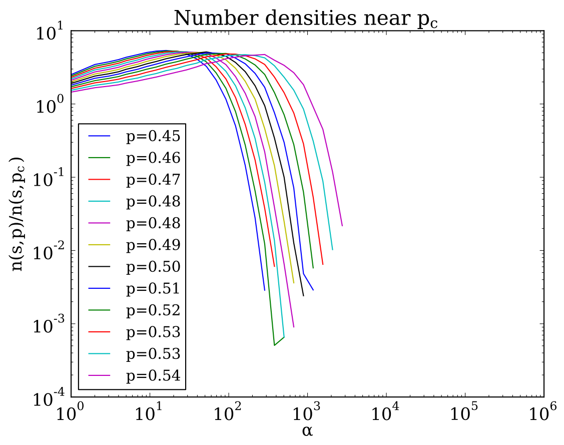

FIG. 13. $n(s, p)/n(s, p_c)$ ratio as a function of s for different p for $L = 1024$.

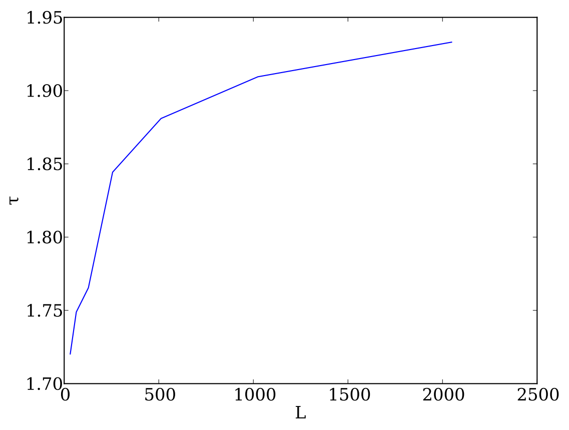

FIG. 12. τ fetched as incline from the different plots in figure 11 as a function of system size L.

close to where it goes to zero could therefore be a good estimate of s_ξ. If we do this for multiple systems with different p, we may finally plot s_ξ as a function of $|p - p_c|$. This is shown in figure 14 where $L = 1024$ has been used.

Taking the loglog of this plot and finding the incline lets us estimate σ as $-1/\sigma$ of the incline. This was found to be

$$\sigma = 0.394 \tag{15}$$

which is pretty close to the true value of $\sigma = 0.396$. However, choosing a different set of p might yield other results (as found by fellow students) as well as a different choice in system size.

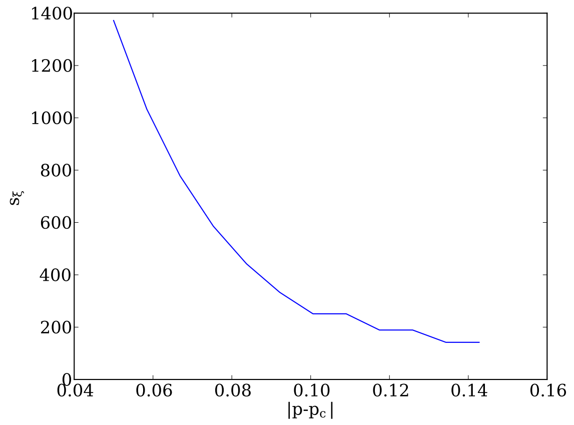

FIG. 14. s_ξ as a function of p for $L = 1024$.

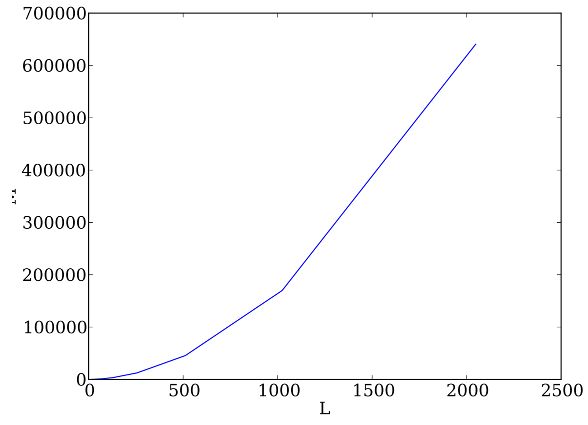

FIG. 15. Mass of the percolating cluster as a function of L.

H. Mass scaling

The mass $M(L)$ of the percolating cluster is shown in figure 15. This is once more an exponential function. Taking the logarithms of M and L and finding the incline gives us the scaling parameter D:

$$M(L) = L^D \tag{16}$$

It was found to be

$$D \approx 1.86. \tag{17}$$

I. Finite size scaling

We define $p_{\Pi=x}$ so that

$$\Pi(p_{\Pi=x}) = x. \tag{18}$$

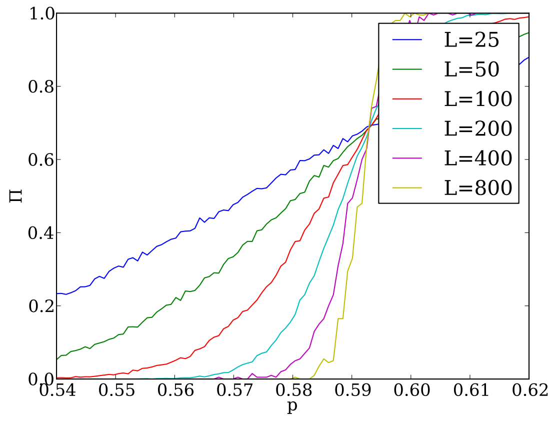

FIG. 16. Probability of spanning cluster as a function of p.

This means that $p_{\Pi=0.3}$ as an example is the value of p for which we have 0.3 probability of getting a spanning cluster.

The next challenge on the list is to find ν, namely scaling parameter ν, related to ξ and ξ_0. According to scaling theory, we have

$$\xi = \xi_0(p_c - p)^{-\nu} \tag{19}$$

We may use this to come to the scaling theory

$$p_{x_1} - p_{x_2} = (C_{x_1} - C_{x_2})L^{-1/\nu} \tag{20}$$

With this, we are able to vary our values for L, make a huge amount of samples and find the probability of having a spanning cluster, Π for different values of p. It is possible by using the halving method or any minimization scheem to find whenever $\Pi = x_1$ and $\Pi = x_2$. I have however done it the simplest possible way, by plotting $\Pi(p)$ for different L in figure 16 and picking the last value where Π is below x_1 and x_2. We chose to use $x_1 = 0.3$ and $x_2 = 0.8$, made a bunch of measurements and found the relation between $p_{\Pi=0.3}$, $p_{\Pi=0.8}$ and L. This proved to be a linear relation, and by taking the incline of the loglog plot, we found

$$\nu = 1.34. \tag{21}$$

This is a pretty decent approximation to the real value.

The good part about this is that we didn't really use any assumption about the value of p_c to get the value of ν. This enables us to find the real value of p_c based on our data. We know that

$$p_{\Pi=x} = p_c + C_x L^{-1/\nu} \tag{22}$$

To measure p_c we may make any assumption about the value of C_x that we want and measure $p_{\Pi=x}$ as a function of $L^{-1/\nu}$. If we look at where the function cuts through the $p_{\Pi=x}$ axis, we get the value of p_c.

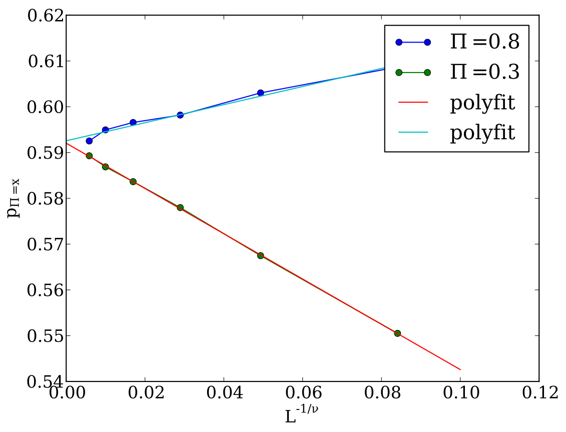

FIG. 17. Plot of the scaling relation $p_{\Pi=x} = p_c + C_x L^{-1/\nu}$ with polyfit applied to find p_c. The value of p_c is found to be $p_c = 0.5922$.

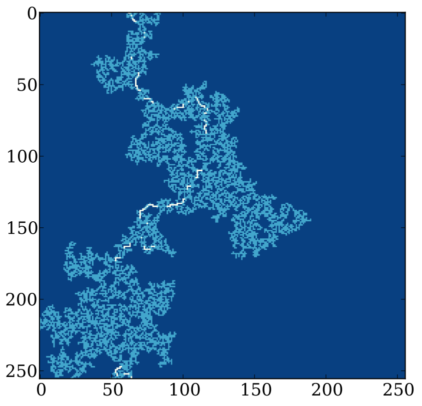

FIG. 18. Spanning cluster shown in light blue and singly connected bonds shown in white. The singly connected bonds are found by the use of one left-turning and one right-turning walker.

This has been done and the result is shown in figure 17. With this we find a value of

$$p_c = 0.5922 \tag{23}$$

which is pretty close to the real value of $p_c = 0.59275$.

J. Singly connected bonds

Using left/right-turning walkers we may locate the singly connected bonds as the sites where both walkers travel through. The result of such a walk is shown in figure 18. By doing this multiple times we are able to find the relation between system size L and the mass of the singly connected bonds. This may be used to estimate

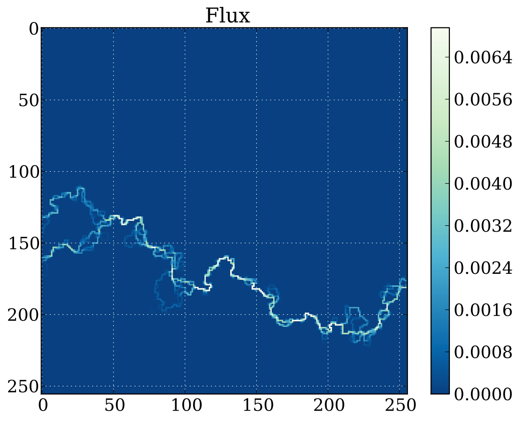

FIG. 19. Currents on the spanning cluster in a 256×256 grid.

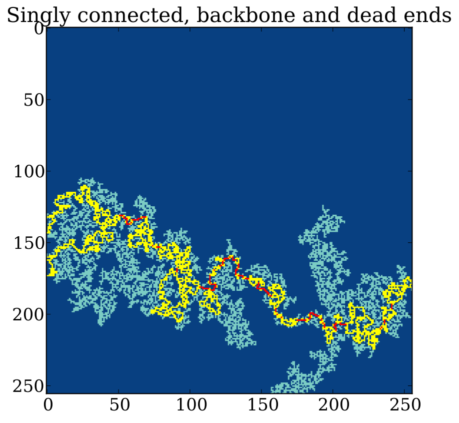

FIG. 20. The singly connected bonds (red), the backbone (yellow) and the dangling ends (green) of the spanning cluster.

the exponent D_{SC} in

$$M_{SC} \propto L^{D_{SC}} \tag{24}$$

which by the usual means of a loglog plot was found to be

$$D_{SC} \approx 0.76. \tag{25}$$

using 5000 samples for $L \in [25, 50, 100, 200, 400, 600]$.

K. Flow on fractals

The currents on the spanning cluster are visualized in figure 19. The singly connected bonds, the backbone and the dangling ends of the spanning cluster are visualized in figure 20. The conductivity is found by the same script by finding the flow out of the spanning cluster through the last column in the matrix and dividing this by the total pressure difference. By finding the conductivity of

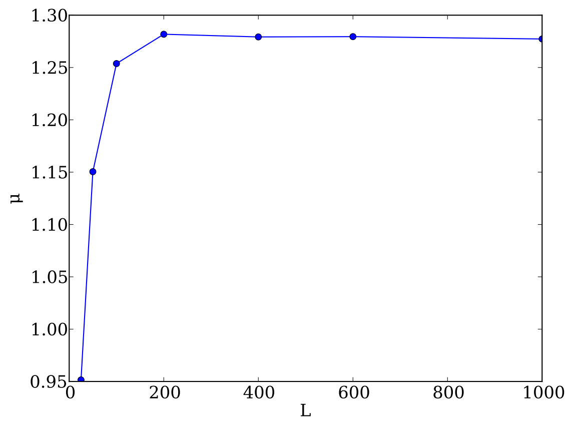

FIG. 21. The exponent μ as a function of system size L.

the cluster at $p = p_c$ for different values of L, we find

$$\zeta_R = 0.977 \tag{26}$$

The dimensionalities are found to be

$$D = 1.87 \tag{27}$$

$$D_{SC} = 0.79 \tag{28}$$

$$D_{BB} = 1.61 \tag{29}$$

$$D_{DE} = 2.09 \tag{30}$$

where we have applied the usual method of creating a loglog plot and finding the incline.

To find the exponent μ, we have also measured the relation between $|p - p_c|$ and C. We are hoping to find a relation like

$$G \propto |p - p_c|^{\mu} \tag{31}$$

By the usual means of a loglog plot, this was measured for different values of L. This gives different values of μ for different L, but a plot of μ against L in figure 21 shows that there is a convergence towards the value of

$$\mu = 1.28. \tag{32}$$

This plot was generated using p-values in the range $p \in [0.62, 0.85]$ and 600 samples for each combination of p and L for $L < 300$ and 60 samples for $L \geq 300$.

This might indicate that the asymptote is going towards a value just above 1 or exactly equal to 1.