FYS4460
PROJECT 3:
PERCOLATION

ALEXANDER HAROLD SEXTON

APRIL 30, 2020

1. Percolating Clusters

Given a matrix of sites, which are either set or not set, we are interested in characterizing the clusters of sites in the matrix. A cluster is said to be spanning if it spans from one side of the system to the opposite side. A quantity of interest is the probability for a system to contain one or more spanning clusters. If it contains at least one such cluster, we say that the system is percolating. For a system of size L and probability p for a site to be set, we characterize this as the percolation probability $\Pi(p, L)$. Another quantity of interest is the density of the spanning clusters in the system, $P(p, L)$. In a two-dimensional system it is defined as

$$P(p, L) = \frac{M_S}{L^2}, \tag{1}$$

where M_S is the number of set sites, and we measure it by simply counting the number of sites in each spanning cluster and divide by the total number of sites. Using the following function, we can calculate both the spanning cluster density and percolation probability for different system sizes and probabilities.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as sp
from skimage import measure

def spanning_cluster_density(m):
    nx, ny = m.shape
    labels, n_features = sp.measurements.label(m)
    regions = measure.regionprops(labels)
    density = 0
    for region in regions:

        x_start, y_start, x_stop, y_stop = region.bbox
        dx = x_stop - x_start
        dy = y_stop - y_start

        if dx == nx or dy == ny:
            density += region.extent

    return density
```

This function takes a binary matrix and uses the magic of the scipy library to identify the nearest neighbours of each site labeled 'true', which then makes up a cluster. By checking the bounds of each cluster, we can simply compare it to the size of the system in each direction, in which case the two are equal, the cluster is spanning, and we save its extent.

We attempt to determine $P(p, L)$ and $\Pi(p, L)$ for two-dimensional systems of size $L \in [2, 4, 8, 16, 32, 64, 128]$ for values of $p \in [0.1, 0.9]$. Since we are determining these quantities stochastically, we calculate each permutation $n = 1000$ in order to get somewhat reliable results. They are shown in figure 1.1 We see from the figure that $P(p, L)$ has a linear behaviour for the smallest systems, and that

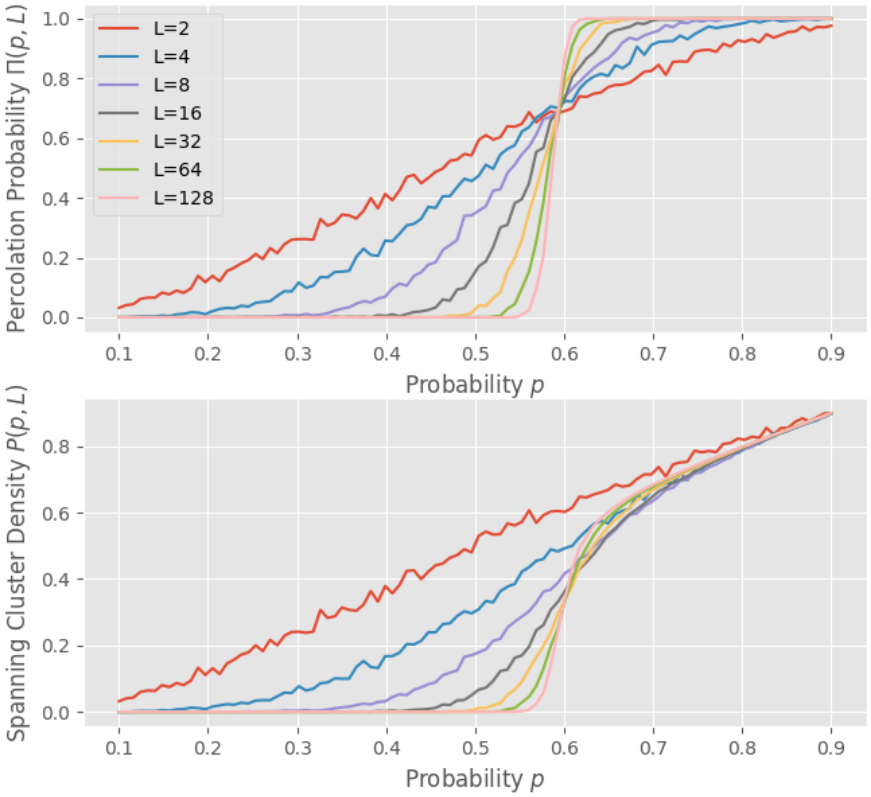

FIGURE 1.1. Percolation probability spanning cluster density (top) and (bot) for different system sizes ($n = 1000$ iterations).

it approaches a step function around the critical point, which looks to be at $p_c \approx 0.56$ for the larger systems. The spanning cluster density exhibits similar behaviour for the small systems, while the larger ones are only linear after $p > p_c$. We also notice that the results for the smaller systems are a lot more noisy than for the larger ones.

2. Determining the exponent of power-law distributions

We generate an array of random numbers following an exponential distribution of some form, in order to develop tools to analyze such distribution. The random numbers are uniformly distributed in the interval $z \in [0, 1]$, and raised to the -2 power.

```
1    import numpy as np
```

```
    z = np.sort(np.random.random(int(1e6))**(-2))

```

The sorting is done so that we may find the cumulative distribution function of these numbers $P(Z > z)$, which is nothing but the integral of the actual distribution $f_z(z)$. We also know that the distribution is of the form $f_z(z) \propto z^\alpha$, so by differentiating $P(Z > z)$ wrt. z and find it's slope, we find the approximate value α, yielding the approximate distribution of the random numbers. We do this with the following script and show the results in figure, where we found $\alpha = -1.499$.

```
import numpy as np
def compute_distribution_function(z):
    n = len(z)
    P = np.linspace(1, n + 1, n)/n
    f = np.diff(P) / np.diff(z)
    alpha, _ = np.polyfit(np.log(z[1:]), np.log(f), deg=1)
    return P, f, alpha
```

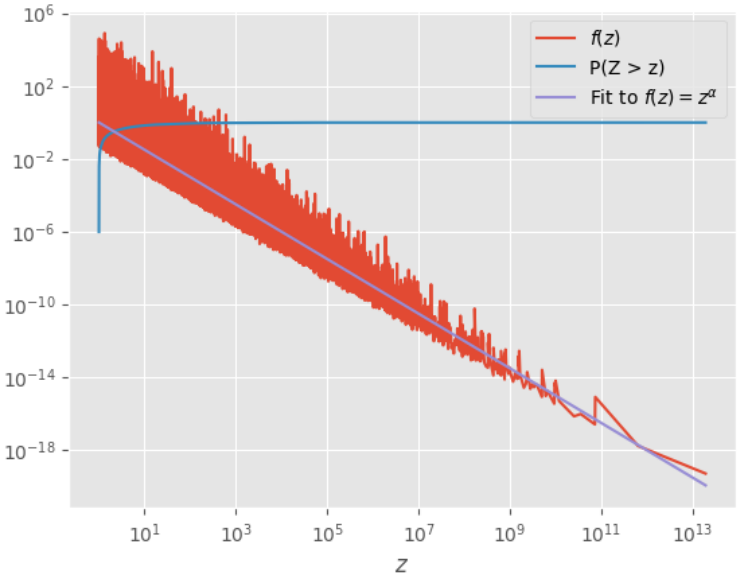

FIGURE 2.1. Log-log plot of the cumulative distribution $P(Z > z)$, the probability distribution $f(z)$ and the fitted curve $f(z) = z^\alpha$.

Looking at the figure, it may seem that there are exponentially many more of the small values than the larger ones. We therefore create a logarithmic binning function, which creates bin sizes which are exponentially increasing, taking care to divide $f(z)_i$ by Δz_i in each case. In figure 2.2 we show this, using $f(z)$ as an example, with base 10 log-binning. The script which performs the binning is listed below.

```
import numpy as np
def log_bin(z, n_bins=10, base=10):
    log_max = np.ceil(np.max(np.log(z))/np.log(base))
    log_bins = np.logspace(0, log_max, n_bins)
```

```
    log_hist, _ = np.histogram(z, bins=log_bins)
    dz = np.diff(log_bins)
    log_bins = 0.5*(log_bins[1:] + log_bins[:-1])
    log_hist_normed = log_hist / dz

    return log_bins, log_hist_normed
```

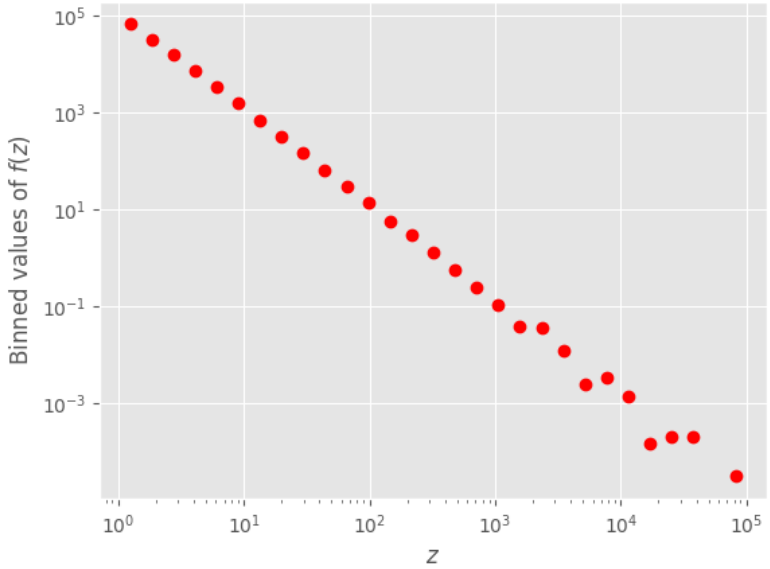

FIGURE 2.2. Log-log plot of $f(z)$ using exponentially increasing bin sizes if log_{10}.

3. Cluster Number Density

Using the two-dimensional system we created in the first section and the method from the previous section of measuring probability densities of rare events, we will now do a numerical computation of the cluster number density $n(s,p)$. The cluster number density is defined as

$$n(s,p) = \frac{N_s}{ML^d}, \tag{2}$$

where N_s is the total number of clusters of size s, L^d is the size of the system, and M is the number of measurements we do. Since occurrences of specific cluster sizes are very rare for the biggest clusters, we will use exponential binning, which means we are in fact measuring

$$\overline{n(s_i,p)}\Delta s_i = \frac{N_i}{ML^d}, \tag{3}$$

and must therefore take care to divide by the bin size Δs_i. We estimate $n(s,p)$ for a two-dimensional system with sides $L = 200$ with values of p approaching p_c from above and below. The results are shown in 3.1, and we have used the following algorithm in producing them.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as sp
```

```python
from skimage import measure

def cluster_number_density(L, p, n_samples, n_bins, logbase=10):
    areas = []
    for i in range(n_samples):
        z = np.random.random((L, L))
        m = z<p
        labels, n_features = sp.measurements.label(m)
        regions = measure.regionprops(labels)
        for region in regions:
            x_start, y_start, x_stop, y_stop = region.bbox
            dx = x_stop - x_start
            dy = y_stop - y_start
            if dx != L and dy != L:
                areas.append(region.area)

    s, N_s = log_bin(areas, n_bins=n_bins)
    #division by delta s is done in the function log_bin
    nsp = N_s/(n_samples*L**2)
    idx = np.where(nsp > 1e-15)[0]
    return s[idx], nsp[idx]
```

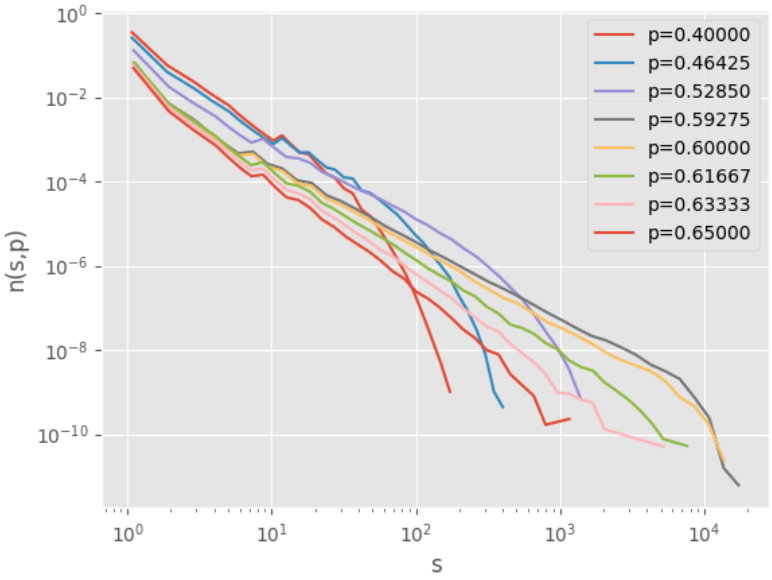

FIGURE 3.1. Plot of $n(s,p)$ of a 2d system with sides $L = 200$ for different values of p.

For the systems where $p < p_c$, there is a power-law behaviour up until a critical point, where it drops off rapidly. We see that this critical point gets larger and larger for p closer to p_c. For values larger than p_c, the power-law stays true for all cluster sizes, and does not have this characteristic cut-off point. Due to the exponential power-law behaviour, it is reason to believe that the cluster number density should follow $n(s,p) \propto s^{-\tau}$. In figure 3.2 we again plot $n(s,p)$, this time for varying system sizes at $p = p_c$. We find an approximate value for $\tau = 1.957$ by finding the slope of $\log_{10} n(s,p)$ vs $\log_{10} s$, whereas the "true" value for this parameter is $187/91 \approx 2.055$. We are

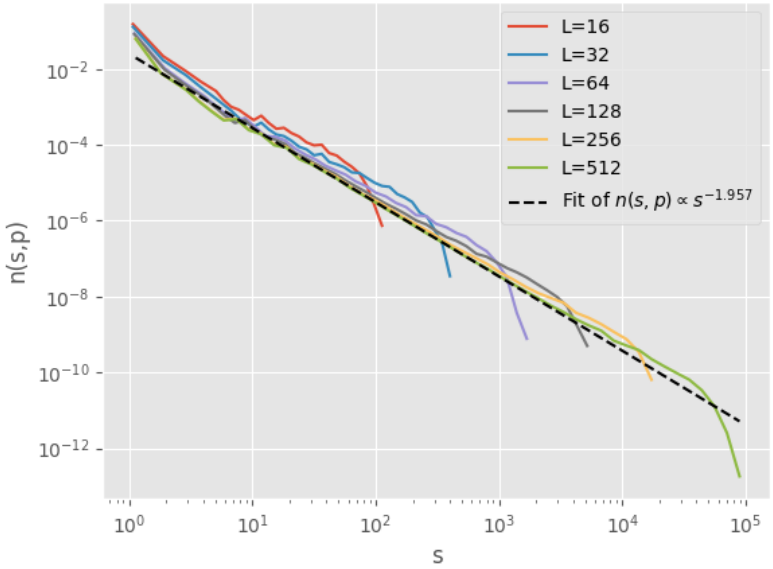

FIGURE 3.2. Log-log plot of $n(s,p)$ for varying L. We also show the fitted exponent τ in the power-law behaviour of the curves.

now interested in finding the scaling of the characteristic cluster size s_ξ, which corresponds to where $n(s,p)$ begins to drop off rapidly. This quantity is given by

$$s_\xi \propto |p - p_c|^{-1/\sigma}, \tag{4}$$

where the "true" value of σ in two dimensions is $\sigma = 36/91 \approx 0.395$. We define s_ξ to be at the point where

$$\frac{n(s,p)}{n(s,p_c)} = F(s/s_\xi) = 0.5, \tag{5}$$

at which we simply read off the corresponding s-value from the $x - axis$. We can then measure σ for our two-dimensional system by doing a linear fit on values s_ξ over a range of values $p < p_c$. With a system of size $L = 2$ figures 3.3 and 3.4 show the results of this, using $n = 1000$ samples, where we can see that s_ξ clearly follows a power law behaviour. From these data we find the exponent $\sigma = 0.462$.

In order to get some more insight into the form of the scaling function $F(s/s_\xi)$, we write it on the form

$$n(s,p) = s^{-\tau} F(s/s_\xi) = s^{-\tau} F((p - p_c)^{1/\sigma} s). \tag{6}$$

By now plotting $s^\tau n(s,p)$ as a function of $s|p - p_c|^{1/\sigma}$ we should get the data-collapse of the cluster number density, which should give the scaling function $F(x)$. Using our previously found values of τ ans σ, figure 3.5 shows the result of this, where we see that the data forms the approximate curve $F(x)$.

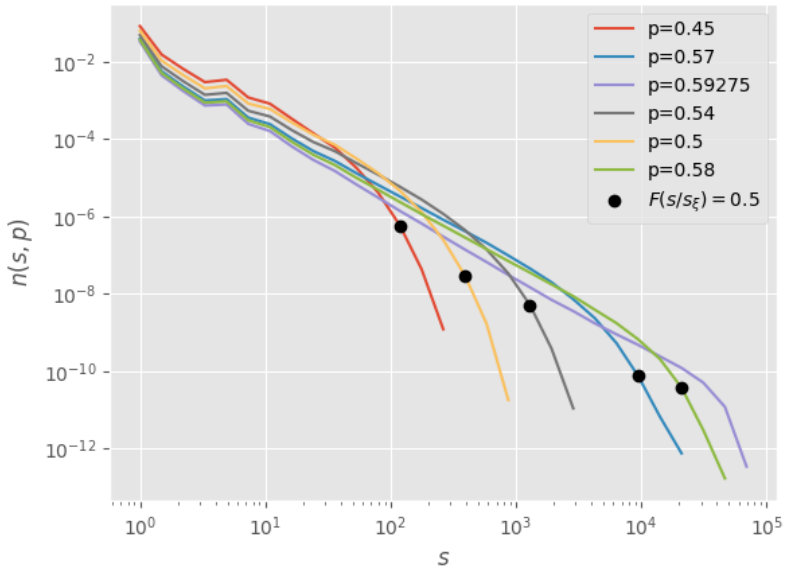

FIGURE 3.3. Plot of the cluster number density for different values of p approaching p_c, with corresponding approximated values for the characteristic cluster sizes s_ξ.

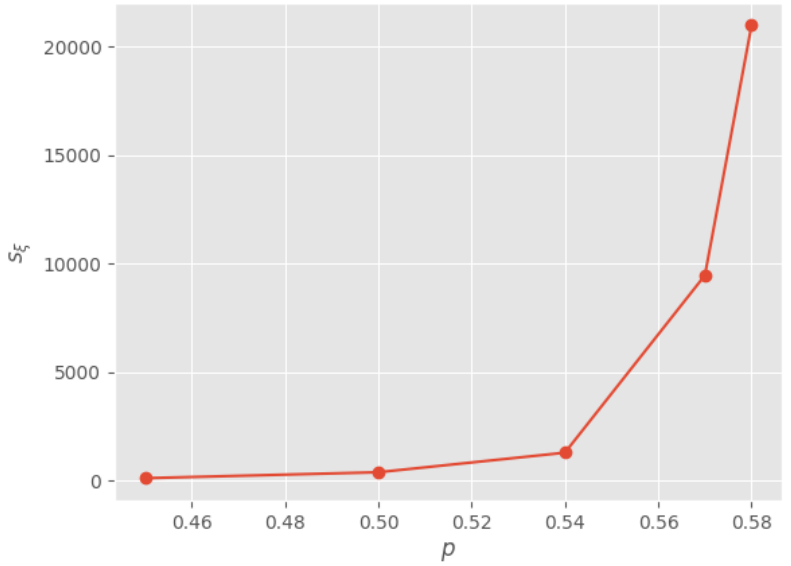

FIGURE 3.4. Plot of the characteristic cluster sizes s_ξ for differing values of p.

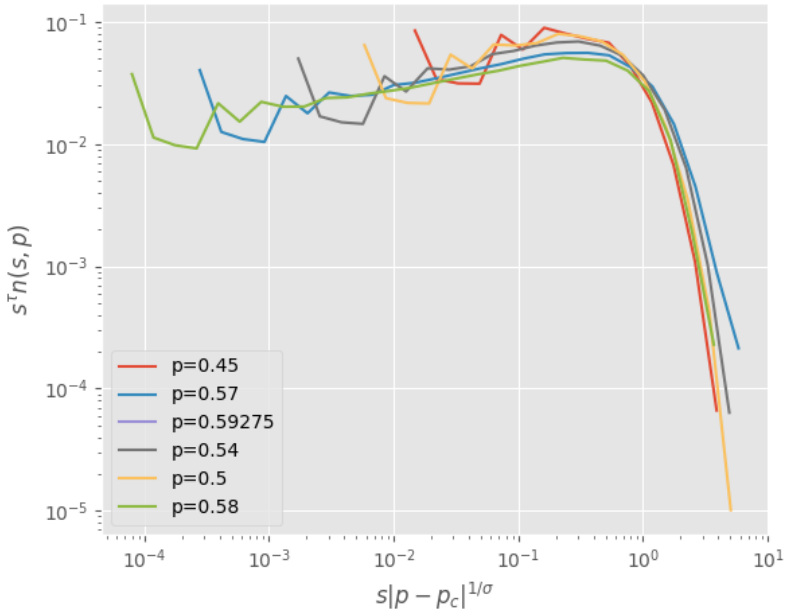

FIGURE 3.5. Data-collapse of $n(s,p)$

4. Mass of the Percolating Cluster

We now wish to find the mass of the percolating cluster $M(L)$ at $p = p_c$. When $L \ll \xi$, we have the relation

$$M(L) \sim L^D. \tag{7}$$

By measuring the mass of the percolating cluster in different system sizes, we can find this exponent D by fitting the data to

$$\log M(L) = log(C) + D \log L. \tag{8}$$

We do this for $L \in [4, .., 11]$ with $n = 100$ samples for each system, and find the exponent $D = 1.8853$. The true value in two dimensions is $D = 91/48 \approx 1.8958$. The results are also shown in figure 4.1.

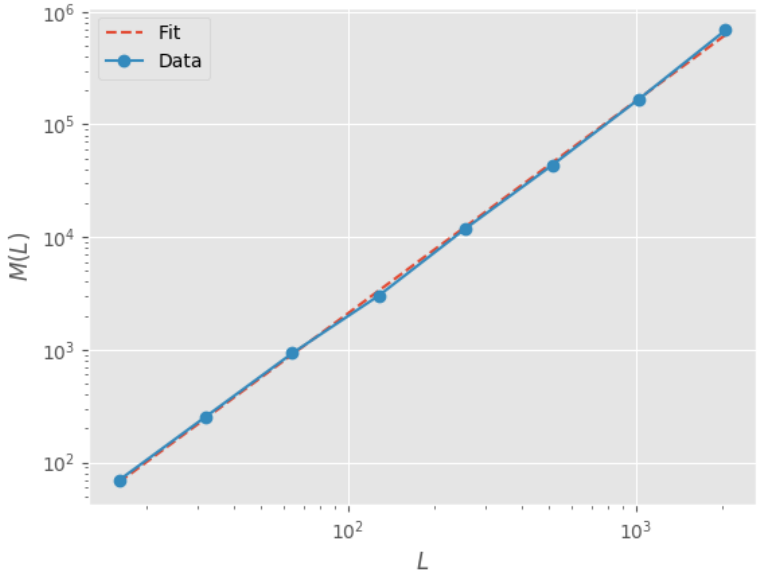

FIGURE 4.1. Mass of the percolating cluster at $p = p_c$ for varying 2d systems.

5. Finite Size Scaling

We will now use a finite-scaling ansatz to find estimates of ν and p_c of a system of size L. Firstly we define $p_{\Pi=x}$ such that $\Pi(p_{\Pi=x}) = x$, that is, the inverse of the percolation probability. By simple interpolation, we can then find the value $p_{\Pi=x} = \Pi^{-1}(x)$, which we do for $x = [0.3, 0.8]$ and $L = [25, 50, 100, 200, 400, 800]$. The results are shown in figure 5.1, where we plot $\Pi(p, L)$ and $p_{\Pi=x}$. We see that $p_{\Pi=0.3}$ and $p_{\Pi=0.8}$ approaches what is presumably p_c.

Using the above results, we can use a result of the scaling ansatz to find an estimate of p_c:

$$p_x - p_c = C_x L^{-1/\nu}. \tag{9}$$

This requires, however, that we know the value ν. We first find ν by subtracting the above equation from itself with a different value x, yielding

$$dp = p_{x_1} - p_{x_2} = (C_{x_1} - C_{x_2})L^{-1\nu}, \tag{10}$$

then find the exponent $-1/\nu$ by fitting a line to the log-log plot of dp. By this method, we find it to be $\nu = 1.405$ (the true value is $nu = 4/3 \approx 1.333$). This result could easily have been improved, by either using a higher resolution grid of p values when calculating $\Pi(p, L)$, using more samples in the calculation of $\Pi(p, L)$, or simply by using a more intelligent interpolation method.

We now use the exact value $\nu = 4/3$ in equation 9, and plot p_x as a function of C_x. Fitting to this line, the y-intercept will be our estimate for p_c. The result is shown in figure 5.2, where we find the approximate value of $p_{c_{x=0.3}} = 0.5925$ and $p_{c_{x=0.8}} = 0.5936$ respectively, which is not far off from the true value $p_{c_{true}} = 0.59275$.

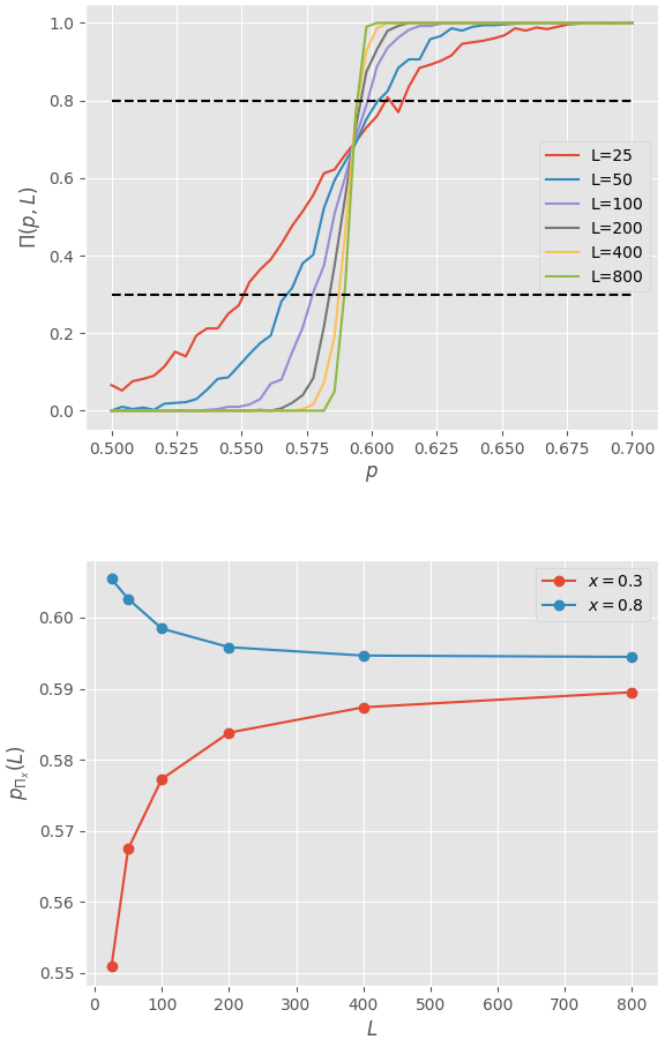

FIGURE 5.1. Plots of $\Pi(p, L)$ and $p_{\Pi=x}$ for varying system sizes.

By rearranging the terms in equation 9, we have that

$$(p_x - p_c)L^{-1/\nu} = C_x = \phi^{-1}(x), \tag{11}$$

where $\phi(x)$ is the scaling function of $\Pi(p, L)$, which from the scaling ansatz is defined as

$$\phi(L^{1/\nu}(p_x - p_c)) = \Pi(p, L). \tag{12}$$

The above equation is nothing but equation 11 where we have applied the function $\phi(x)$ on both sides of the equality. We can find the shape of $\phi(x)$ simply by plotting $\Pi(p, L)$ as a function of $(p_x - p_c)L^{-1/\nu}$, creating a data-collapse of $\Pi(p, L)$. Using the data from the previous exercise, we plot this for the various L, and the results are shown in 5.3. We can see that the curves all

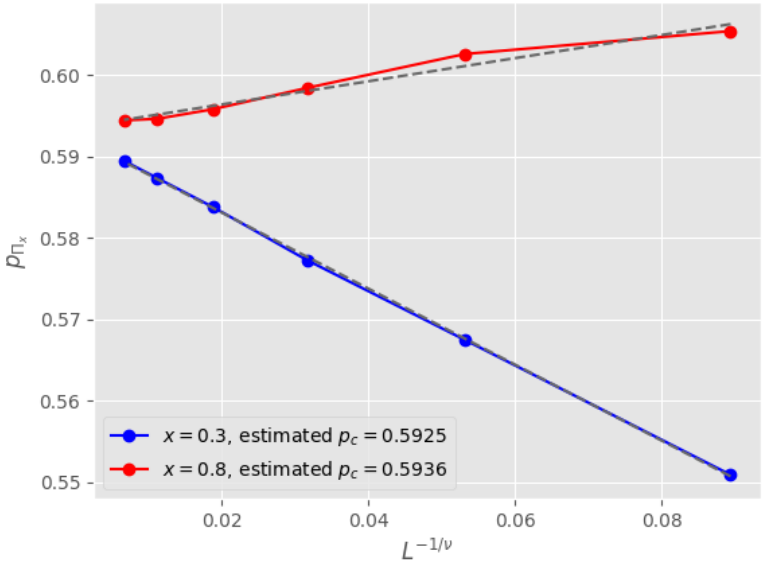

FIGURE 5.2. Plot of p_{Π_x} vs $L^{-1/\nu}$ for $x = 0.3$, $x = 0.3$ and varying L. The y-intercept corresponds to the estimated value p_c.

fall onto one common curve, which demonstrates that the measured data is consistent with the scaling ansatz.

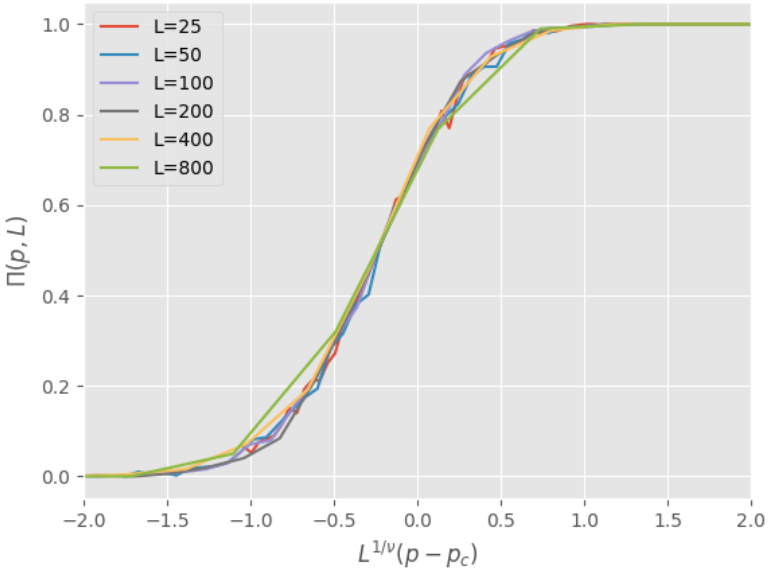

FIGURE 5.3. Plot of $\Pi(p, L)$ as a function of $L^{1/\nu}(p - p_c)$, yielding the shape of the scaling function $\phi(x)$.

6. Singly Connected Bonds

We now wish to visualize the sub sets of the spanning cluster. The geometry of clusters is by part characterized by its *singly connected bonds*, which is a path that connects one part of the cluster to the other, and if removed, the cluster is no longer percolating. We also have the *backbone* of the cluster, which is the union of all paths that lead from one side of the cluster to the other. The remaining sub set is the *dangling ends*, which are the paths that lead to the "dead ends" of the cluster, and do not contribute to for instance flow through the cluster. In order to visualize this, we firstly generate a spanning cluster at $p = p_c$ and use the provided "walk.py" script. The algorithm deploys a left and right walker which both start at one of the clusters ends and attempts to reach the other side. For each step the left walker turns left, if possible, else it tries the other directions. Similar for the right walker, but will turn right as it's first choice. This continues until they reach the opposite side of the cluster. For each site that is visited by the walkers, we increment a counter at that specific position, which can be plotted in order to visualize the singly connected bonds and backbone of the spanning cluster, as shown in figure ref. Brighter colors represent sites which are visited more frequently.

We now wish to find the mass of the singly connected bonds M_{SC} as a function of system size L, at $p = p_c$. We assume that also this quantity follows a power law,

$$M_{SC} \propto L^{D_{SC}}. \tag{13}$$

We generate spanning clusters for system sizes of $L = 2^k$ for $k \in [4, 10]$ and measure M_{SC} for each one, averaging over 100 samples for each system size. The results are shown in figure 6.2, where we see that M_{SC} indeed follows a power law. Doing a linear fit on the simulation results, we also find the approximation to the exponent $D_{CS} = 0.7822$ (true value $D_{SC} = 3/4$).

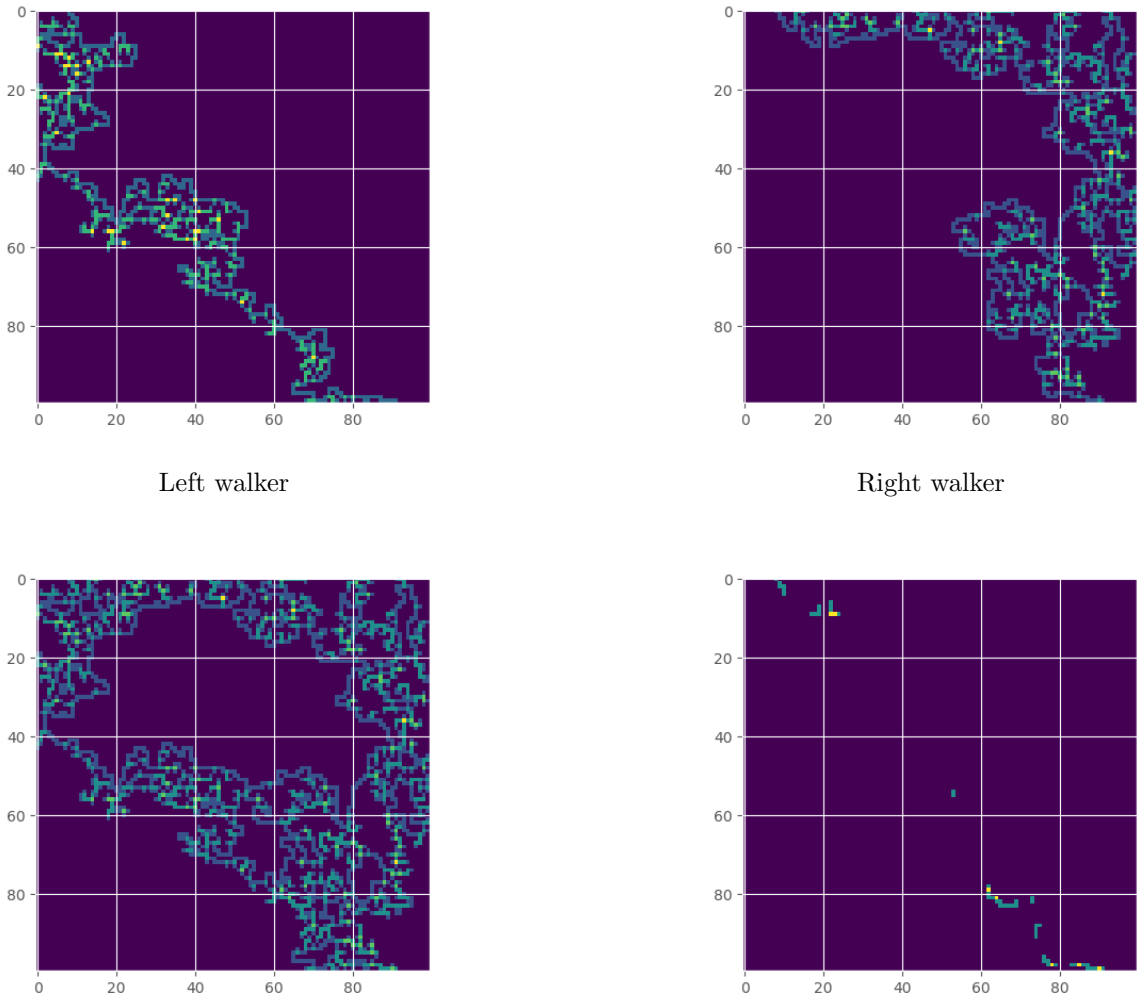

FIGURE 6.1. Visualization of the paths of the right and left walkers.

We also look at the behaviour of the density of the singly connected bonds, which is given by

$$P_{SC} = \frac{M_{SC}}{L^d} \propto \frac{L^{D_{SC}}}{L^d}. \tag{14}$$

We wish to find the behaviour of P_{SC} as a function of $p - p_c$ for $p \geq p_c$. Since the singly connected bonds are the only connecting sites between two regions of the spanning cluster, we expect P_{SC} to decrease as p increases, which is confirmed by figure 6.3. It may also seem that the density follows a power law behaviour $P_{SC} \propto (p - p_c)^x$. To investigate this, we plot $\log P_{SC}$ as a function of $\log(p - p_c)$ and look for the exponent x by doing a linear fit. The results are shown in figure 6.4, where we plot this relation for a system of size $L = 512$, and it suggests that it does indeed follow a power law relation.

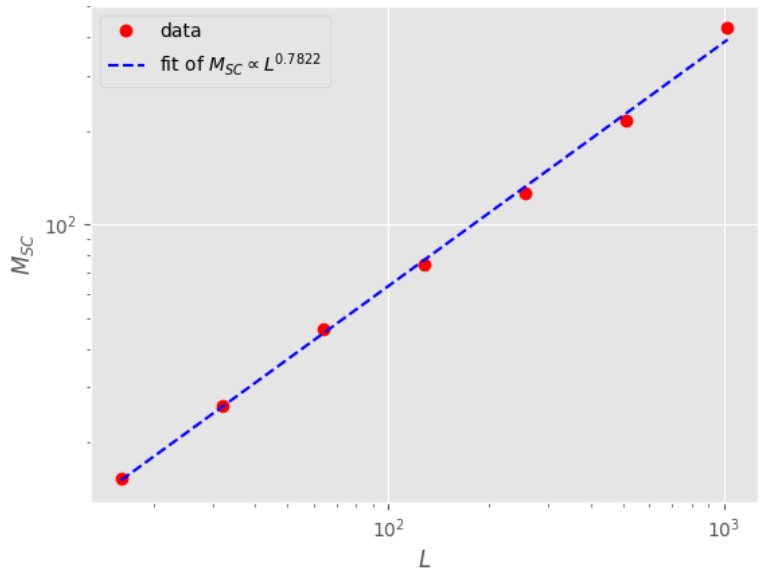

FIGURE 6.2. Mass of the singly connected bonds as a function of system size.

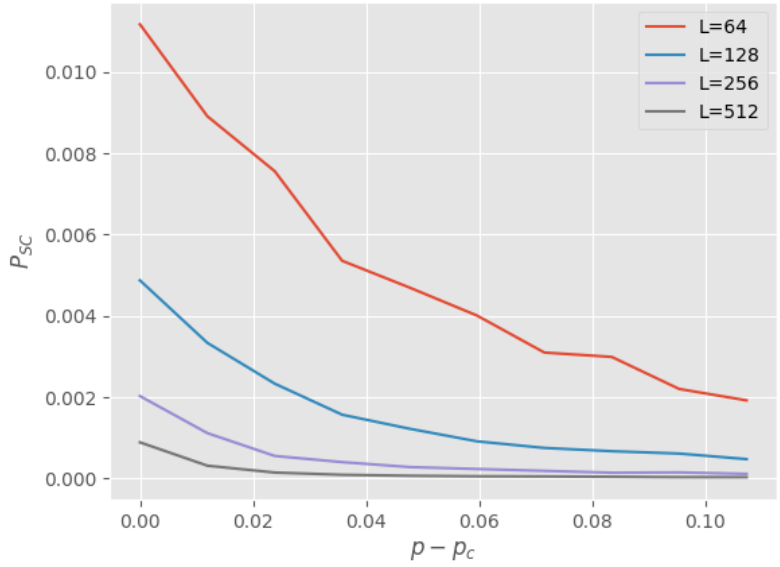

FIGURE 6.3. Plot of P_{SC} as a function of $p - p_c$ for various L.

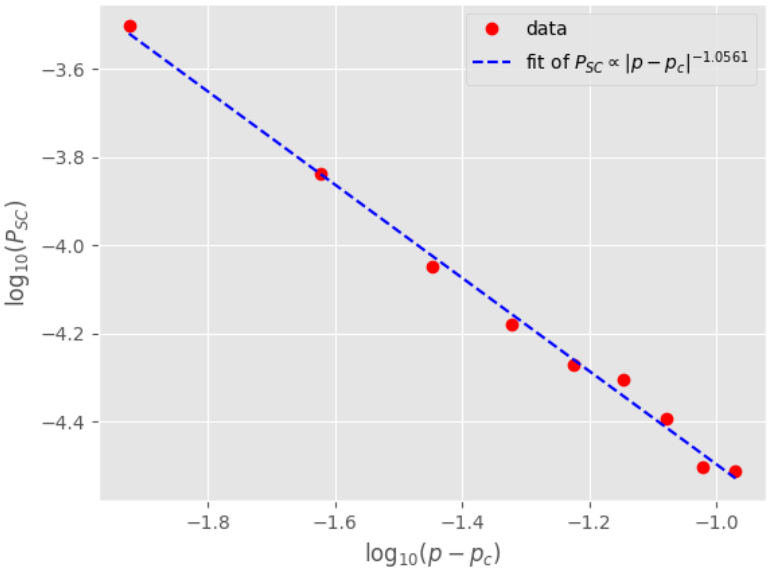

FIGURE 6.4. log-plot of P_{SC} as a function of $p - p_c$ for a system of size $L = 512$.