FYS4460:
UNORDERED SYSTEMS AND PERCOLATION
PERCOLATION TOPICS

ALEXANDER HAROLD SEXTON

JUNE 10, 2020

Contents

1. Topic 9: Algorithms for percolation systems 2
2. Topic 10: Percolation on small lattices 7
3. Topic 11: Cluster number density in 1-d percolation 11
4. Topic 12: Cluster size in 1-d percolation 16
5. Topic 13: Measurement and behavior of $P(p, L)$ and $\Pi(p, L)$ 21
6. Topic 14: The cluster number density 24
7. Topic 15: Finite size scaling of $\Pi(p, L)$ 30
8. Topic 16: Effective percolation threshold 35
9. Topic 17: Subsets of the spanning cluster 39

1. Topic 9: Algorithms for percolation systems

How do we generate a percolation system for simulations? How to analyze and visualize the systems? How to find spanning clusters and measure the percolation probability?

- Percolation systems are generated by defining a 2-d grid of random numbers.
- Define sites to be set if site is below some value p

```
L = 4
p = 0.5
z = np.random.random((L, L))
system = z<p
```

- Visualize system using `plt.imshow()` from `matplotlib`
- Example of this in figure 1.1.

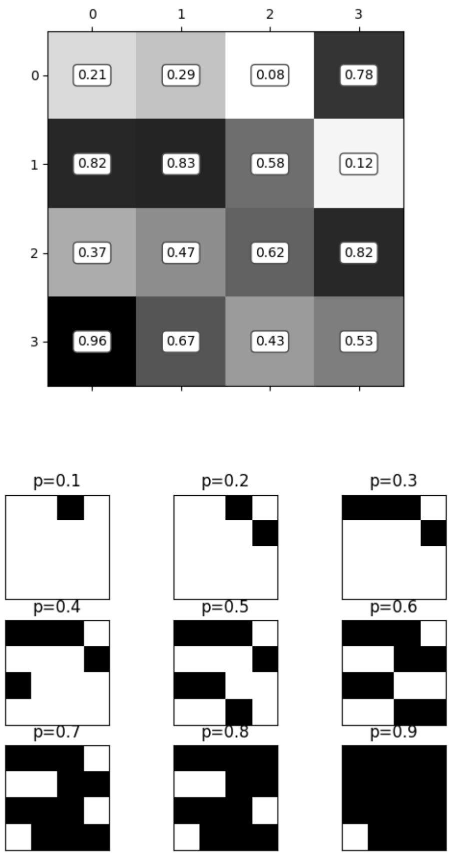


Figure 1.1. Percolation systems with varying p.

- Characterize clusters by which sites are connected
- Next-neighbour connectivity
- Percolating systems
- Use python libraries for analyzing imagery

```python
import scipy.ndimage as sp
from skimage import measure
labels , n_features = sp.measurements.label(system)
```

- Returns a matrix of connected regions which are labeled
- Can be visualized easily as in figure 1.2

```python
L = 10
p = 0.5
z = rand(L,L)
system = z<p
labels, n_features = measurements.label(system)
clusterArea = measurements.sum(system, labels, index=arange(labels.max() + 1))
areaImg = clusterArea[labels]
cmap = plt.cm.viridis
cmap.set_under('black')
imshow(areaImg, origin='lower', cmap=cmap, vmin=0.1, interpolation='none')
colorbar()
show()
```

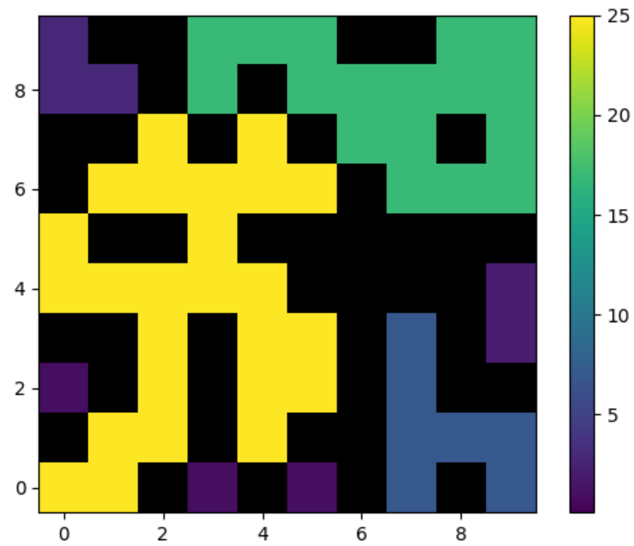


FIGURE 1.2. Visualization of the clusters in the system

- System of size $L = 100$ for p approaching p_c in figure 1.3

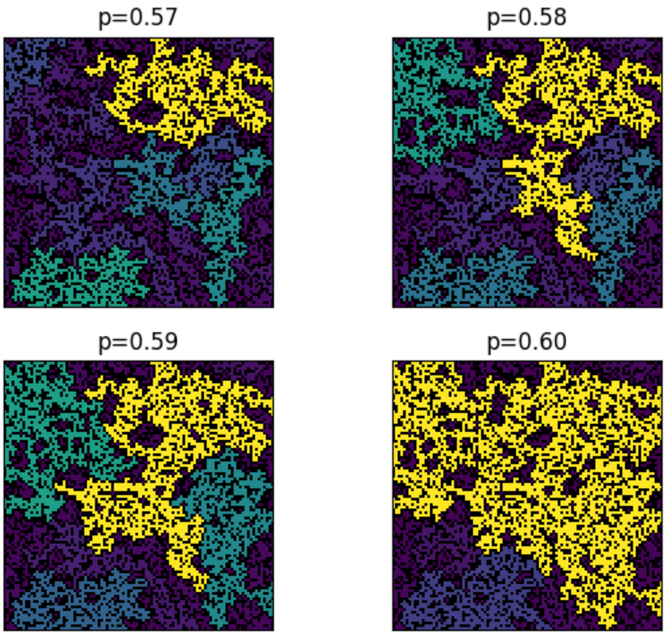


FIGURE 1.3. 100×100 systems with p approaching p_c.

- Use the bounding box of clusters to determine if system is percolating

```
def spanning_cluster_density(m, single_perc = False):
    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
            if single_perc:
                break

    return density
```

- Measure percolation probability by Monte Carlo simulations

```
fig, (ax1, ax2) = plt.subplots(2)
for L in L_arr:
    Pi_arr = np.zeros(len(p_arr))
    density_arr = np.zeros(len(p_arr))
    for p, prob in enumerate(p_arr):
```

```
        for i in range(n_samples):
            z = np.random.random((L,L))
            m = z<prob
            density = spanning_cluster_density(m)
            density_arr[p] += density
            if density > 0:
                Pi_arr[p] += 1

    Pi_arr /= n_samples
    density_arr /= n_samples
    ax1.plot(p_arr, Pi_arr, label=f'L={L}')
    ax2.plot(p_arr, density_arr, label=f'L={L}')
```

- Bonus: this code also computes the density of the spanning cluster. Example in figure 1.4

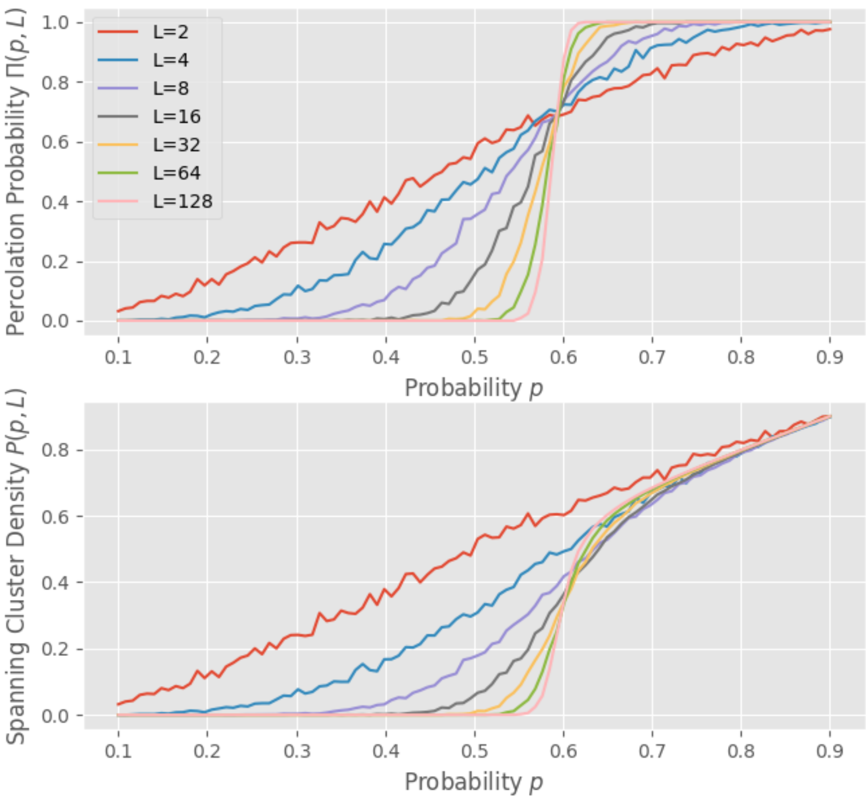


FIGURE 1.4. Spanning cluster density and percolation probability for different system sizes, $n = 1000$ samples.

2. Topic 10: Percolation on small lattices

Discuss the percolation problem on a 2×2 lattice. Sketch $P(p, L)$ and $\Pi(p, L)$for small L. Relate to your simulations. How do you calculate these quantities and how do you measure them in simulations?

- The 2×2 percolation problem is simple enough for us to list the possible outcomes by hand.

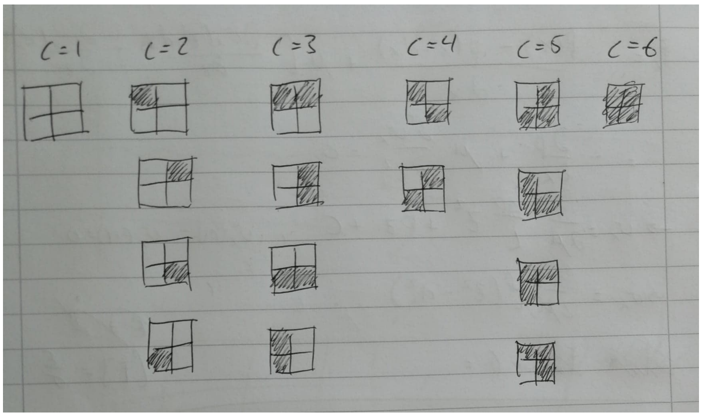


Figure 2.1. The 16 configurations of the 2×2 lattice.

- Can find exact expressions for $\Pi(p, L)$ and $P(p, L)$

- $\Pi(p, L) = \sum_c g_c \Pi(p, L|c) P(c)$

- Probabilities listed in table 2.1

Table 2.1. List of probabilities of the configurations of the 2×2 lattice, and the percolation probability.

c	g	$P(c)$	$\Pi(p, L\|c)$
1	1	$p^0(1-p)^4$	0
2	4	$p^1(1-p)^3$	0
3	4	$p^2(1-p)^2$	1
4	2	$p^2(1-p)^2$	0
5	4	$p^3(1-p)^1$	1
6	1	$p^4(1-p)^0$	1

- This gives $\Pi(p, L=2) = 4p^2(1-p)^2 + 4p^3(1-p) + p^4 = p^2(p-2)^2$

- Density of the spanning cluster can be found from $p = \sum_{s=1}^{\infty} sn(s,p) + P(p, L)$

- Use same method as for the density of the spanning cluster:

$$
\begin{align}
P(p, L=2) &= p - \sum_{s=1}^{4}\sum_{c=1}^{6} sn(s, p|c)P(c) \tag{1}\\
&= p - \left[p(1-p)^3 + p^2(1-p)^2\right] \tag{2}\\
&= p^2(2-p) \tag{3}\\
&\text{(a bit unsure about this result, as we will see)} \tag{4}
\end{align}
$$

- Compare exact expressions to numerical results
- Monte Carlo simulations with $M = 1000$ samples
- Measurement of spanning cluster density: $P(p, L) = \frac{M_s}{L^d}$
- Calculated by the following algorithm

```python
def spanning_cluster_density(m, single_perc = False):
    nx, ny = m.shape
    labels, n_features = sp.measurements.label(m)
    regions = measure.regionprops(labels)
    mass = 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:
            mass += region.extent
            if single_perc:
                break
    density = mass/(nx*ny)
    return density
```

- Plot is shown in figure 2.2.
- Result from larger systems in figure 2.3

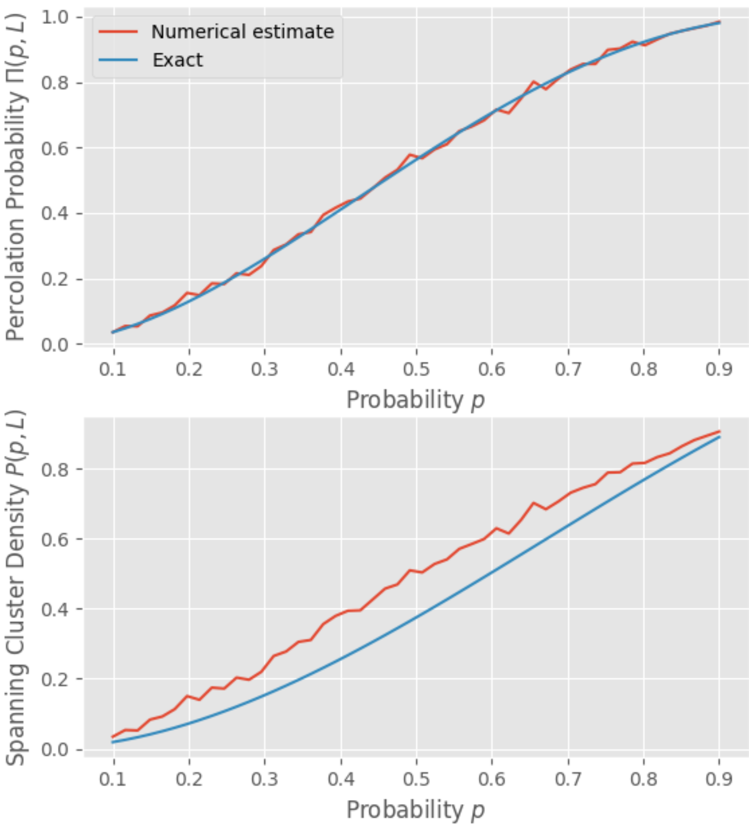


FIGURE 2.2. Density of the spanning cluster and percolation probability of the 2×2 lattice.

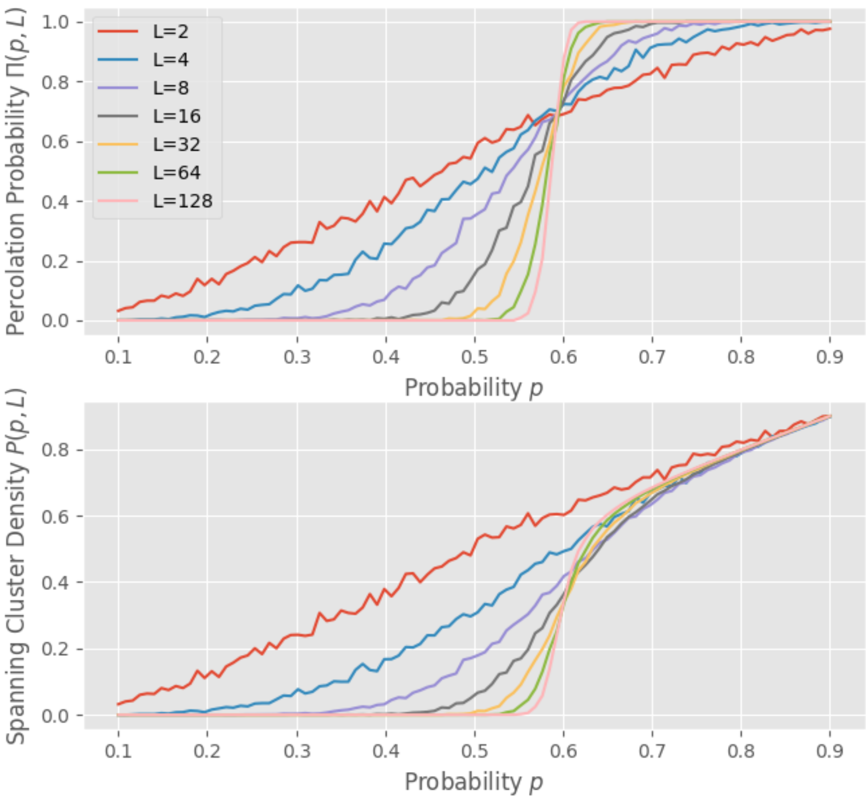


FIGURE 2.3. Density of the spanning cluster and percolation probability of different system sizes.

3. Topic 11: Cluster number density in 1-d percolation

Define the cluster number density for 1-d percolation, and show how it can be measured. Discuss the behavior when $p \to p_c$. How does it relate to your simulations in two-dimensional systems?

- Definition of the cluster number density $n(s, p)$
- Definition of $P = sn(s, p)$
- Useful for characterizing clusters in a system
- 1-d percolation is a simple system, but also a powerful tool
- Cluster number density in 1-d: $n(s, p) = p^s(1 - p)^2$

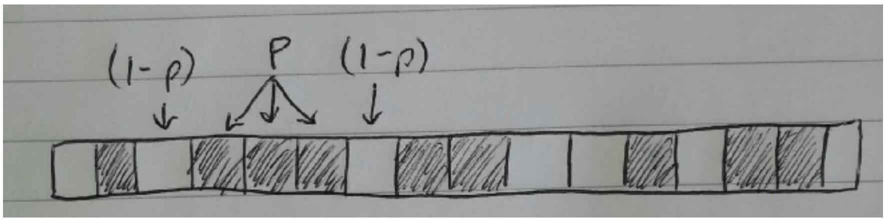


Figure 3.1. Percolation system in 1d.

- Check that it is normalized by calculating $p = \sum_{s=1}^{\infty} sn(s, p) + P$
- Numerical estimates of the cluster number density
- Can also estimate the probability by this method
- Using the above example: $\overline{p} = \sum_s \overline{sn(s, p)} = \sum_s \frac{sN_s}{L^d} = \frac{9}{16}$

Table 3.1. Table of values for the cluster number density from the example in figure 3.1.

s	N_s	$n(s, p)$
1	2	2/16
2	2	2/16
3	1	1/16

- Use of Monte Carlo methods for measuring good estimates: $\overline{n(s, p)} = \frac{N_s(M)}{ML^d}$
- Logarithmic binning

- Algorithm:

```python
def log_bin(z, n_bins=10, base=10, log_max=None):
    if log_max is None:
        log_max = np.ceil(np.max(np.log(z))/np.log(base))
    log_bins = np.logspace(0, log_max, n_bins, base=base)
    log_hist, _ = np.histogram(z, bins=log_bins)
    dz = np.diff(log_bins)
    bin_mid = 0.5*(log_bins[1:] + log_bins[:-1])
    log_hist_normed = log_hist / dz

    return bin_mid, log_hist_normed
```

- Code for measuring $n(s, p)$

```python
def cluster_number_density(L, p, n_samples, n_bins, logbase=10, log_max=None,
    remove_zeros=False):
    areas = []
    for i in tqdm(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, base=logbase, log_max=log_max)
    nsp = N_s/(n_samples*L**2)
    if remove_zeros:
        idx = np.where(nsp > 1e-15)[0]
        return s[idx], nsp[idx]
    else:
        return s, nsp
```

- Behaviour of $n(s, p)$ when $p \to p_c$ in 1-d. Figure 3.2

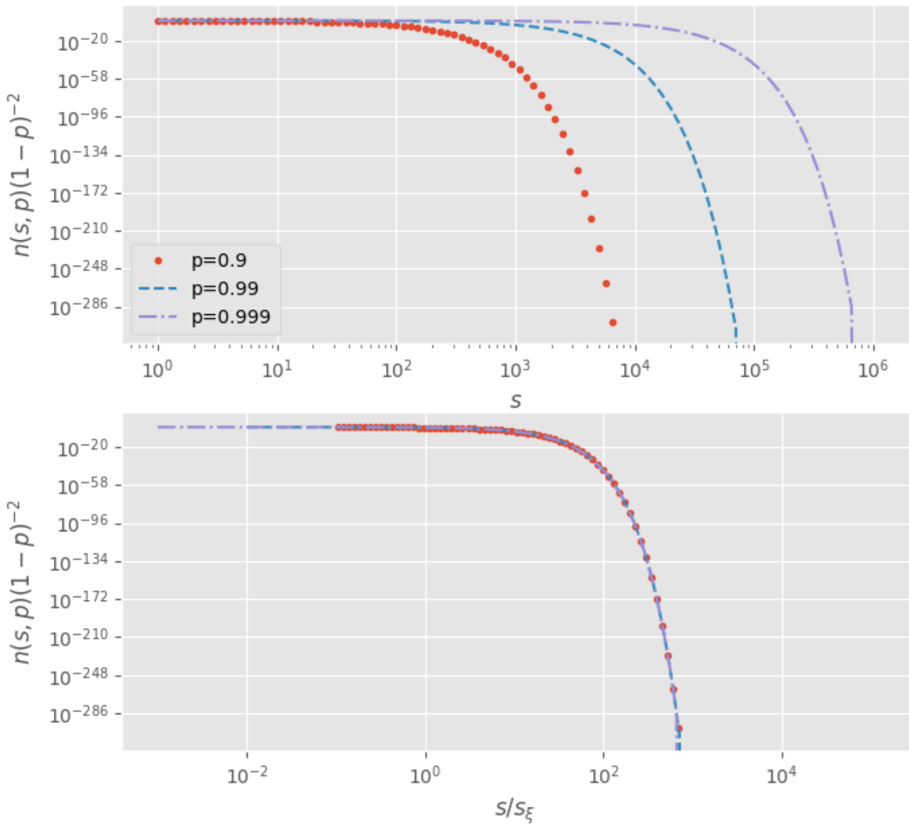


FIGURE 3.2. Plot of $n(s,p)(1-p)^{-2}$ as a function of s and s/s_ξ for a 1d system of length $L = 10^6$.

- Similar behaviour in 2 dimensions. Figure 3.3

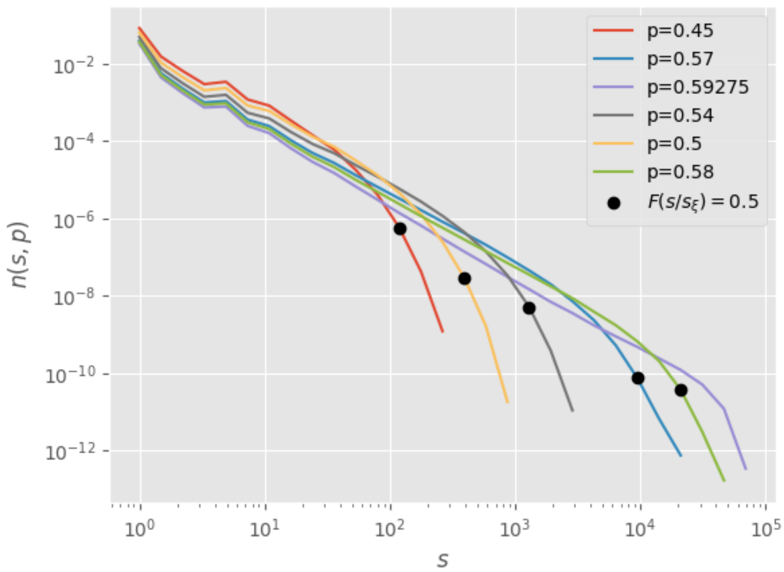


FIGURE 3.3. Cluster number density of a 2d system of size $L = 512$ for values of p approaching p_c.

- Understanding the behaviour by rewriting:

$$n(s,p) = (1-p)^2 p^s = (1-p)^2 e^{s \ln p} = (1-p)^2 e^{-s/s_\xi}, \tag{5}$$

where

$$s_\xi = -\frac{1}{\ln p}. \tag{6}$$

- Exponential power-law behaviour can be seen in the above loglog plot: $n(s,p) \propto s^{-\tau}$
- Estimating the scaling exponent τ
- Results from 2d: $\tau = 1.957$
- "True" value $\tau = 187/91 \approx 2.055$

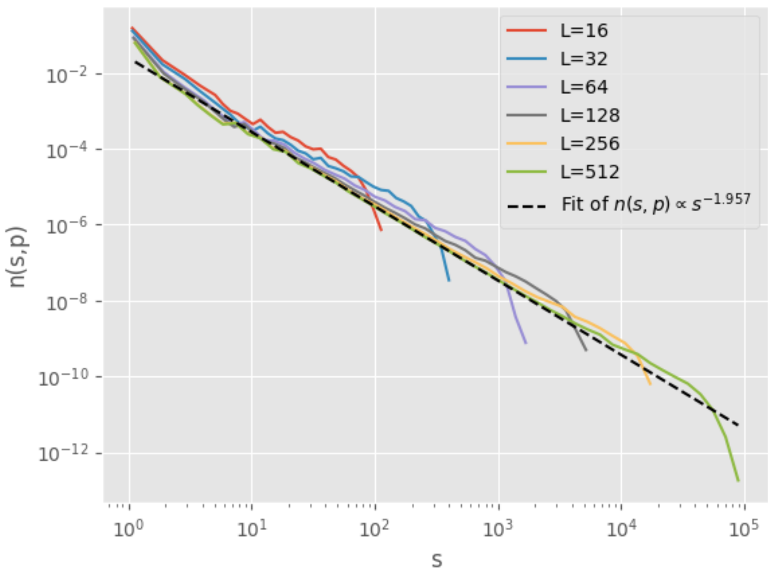


FIGURE 3.4. Cluster number density for varying system sizes at $p = p_c$. We also show the fitted exponent τ of the power law behaviour.

4. Topic 12: Cluster size in 1-d percolation

Introduce the characteristic cluster size for the 1-d percolation problem, and discuss their behavior when $p \to p_c$. Relate to your simulations on two-dimensional percolation

- Cluster number density in 1d: $n(s,p) = (1-p)^2 p^s$

- How does it behave in relation to s when $p \to p_c$? Figure 4.1

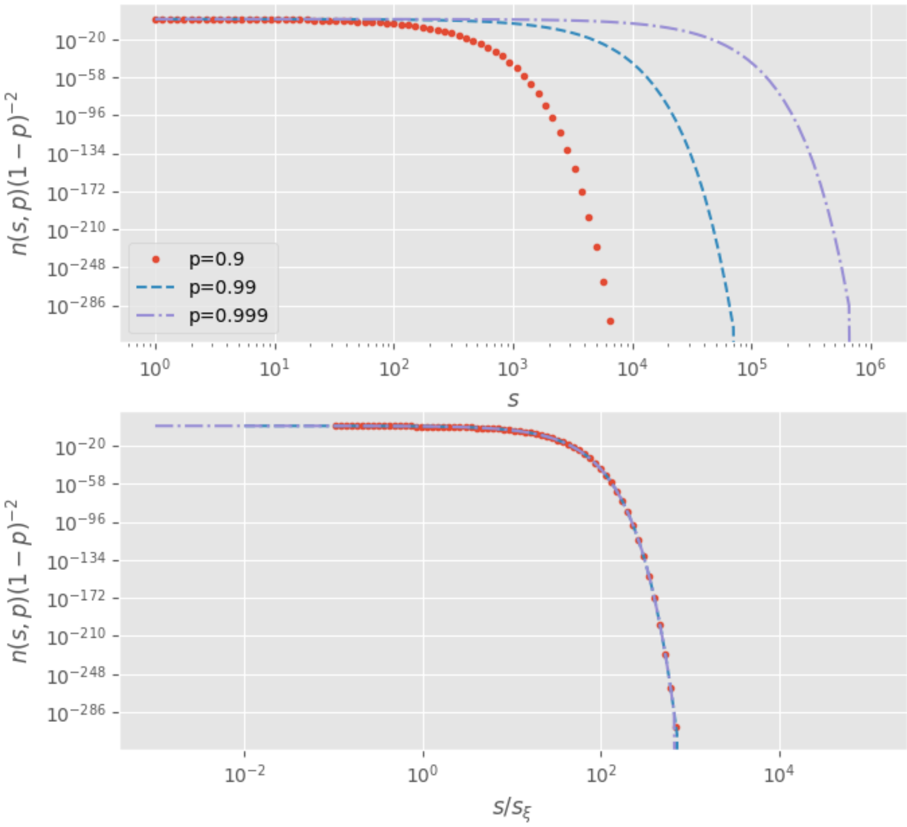


Figure 4.1. Plot of $n(s,p)(1-p)^{-2}$ as a function of s and s/s_ξ in a 1d system of size $L = 10^6$.

- Definition of the characteristic cluster size s_ξ

- Diverges as $p \to p_c$

- Can understand this by rewriting the cluster number density:

$$n(s,p) = (1-p)^2 p^s = (1-p)^2 e^{s \ln p} = (1-p)^2 e^{-s/s_\xi}, \tag{7}$$

where

$$s_\xi = -\frac{1}{\ln p}. \tag{8}$$

- s_ξ causes exponential fall-off of $n(s,p)$

- Also the case in 2d, figure 4.2

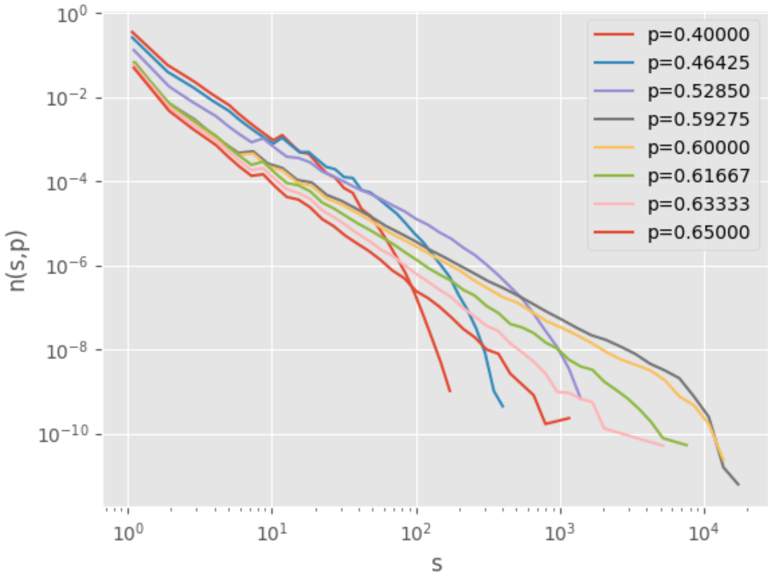


FIGURE 4.2. Plot of $n(s,p)$ for varying p in a 2d system of size $L = 200$.

- Generalizing s_ξ when $p \to p_c$, Taylor expansion of $\ln p \approx -(1-p)$:

$$s_\xi \approx \frac{1}{1-p} = \frac{1}{p_c - p} = |p - p_c|^{-1/\sigma}, \tag{9}$$

- s_ξ marks the cross-over between the two behaviours of $n(s,p)$.

- Characterize s_ξ by measurements of $n(s,p)$

- Can be defined as

$$\frac{n(s,p)}{n(s,p_c)} = F(s/s_\xi) = 0.5. \tag{10}$$

- Use this in simulations of $n(s,p)$ to find σ in 2d, figure 4.3

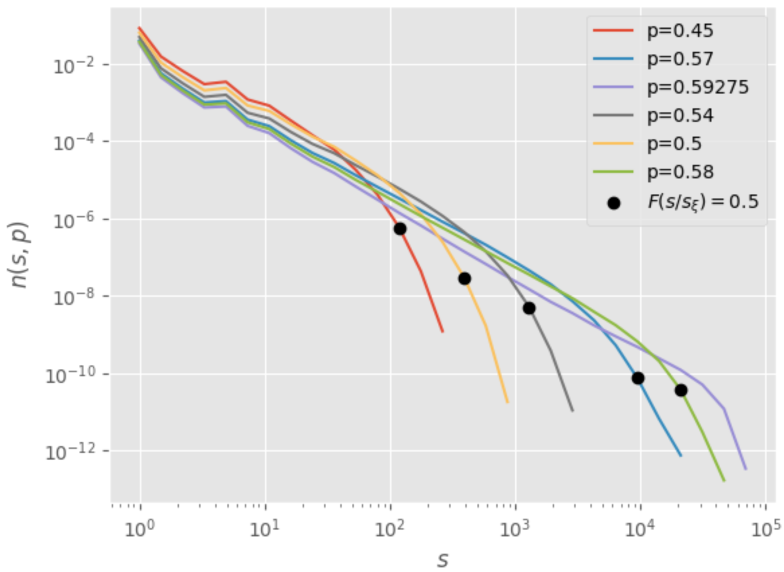


FIGURE 4.3. Cluster number density of a 2d system of size $L = 512$ for values of p approaching p_c, with corresponding values of s_ξ.

```
def log_bin(z, n_bins=10, base=10, log_max=None):
    if log_max is None:
        log_max = np.ceil(np.max(np.log(z))/np.log(base))
    log_bins = np.logspace(0, log_max, n_bins, base=base)
    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

def cluster_number_density(L, p, n_samples, n_bins, logbase=10, log_max=None,
    remove_zeros=False):
    areas = []
    for i in tqdm(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, base=logbase, log_max=log_max)
    nsp = N_s/(n_samples*L**2)
    if remove_zeros:
```

```python
        idx = np.where(nsp > 1e-15)[0]
        return s[idx], nsp[idx]
    else:
        return s, nsp

def estimate_s_xi(L, n_samples, n_bins):
    pc = 0.59275
    eps = 1e-14
    p_vals = [0.45, 0.50, 0.54, 0.57, 0.58]
    logbase = 1.3
    remove_zeros = False
    s, nc = cluster_number_density(L, pc, n_samples, n_bins, logbase, remove_zeros=remove_zeros)
    log_max = np.log(s[-1])/np.log(logbase)
    nonzero = np.where(nc > eps)[0]
    sxi_vals = []
    F_vals = []
    p_array = []
    for i, p in tqdm(enumerate(p_vals)):
        s, n = cluster_number_density(L, p, n_samples, n_bins, logbase, log_max, remove_zeros)
        nonzero = np.where(n > eps)[0]

        temp_sxi = []
        temp_n = []
        for i in range(len(n)):
            if n[i] > eps and nc[i] > eps:
                if n[i]/nc[i] <= 0.5:
                    temp_sxi.append(s[i])
                    temp_n.append(n[i])


        if len(temp_n)> 0:
            idx = np.argmax(temp_n)
            F = temp_n[idx]
            sxi = temp_sxi[idx]
            sxi_vals.append(sxi)
            F_vals.append(F)
            p_array.append(p)

    return sxi_vals, p_array
```

- Linear fit $\rightarrow \sigma = 0.462$
- "True" value: $\sigma = 36/91 \approx 0.395$.
- Plot of s_ξ as function of p in figure 4.4

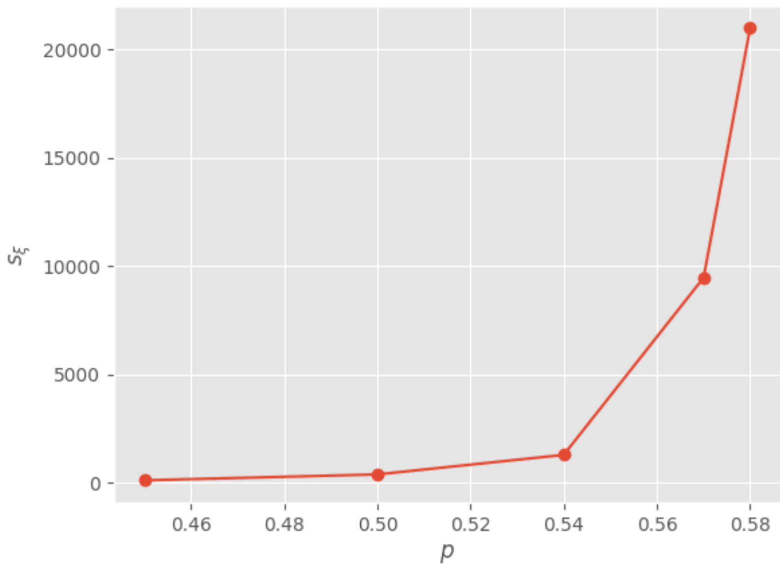


FIGURE 4.4. s_ξ as a function of p for a system of size $L = 512$.

5. Topic 13: Measurement and behavior of $P(p, L)$ and $\Pi(p, L)$

Discuss the behavior of $P(p, L)$ and $\Pi(p, L)$ in a system with a finite system size L. How do you measure these quantities?

- Definition of $\Pi(p, L)$
- Definition of $P(p, L)$
  $P(p, L) = \frac{M_S}{L^d}$
- Expected behaviour of these quantities
- Small vs large systems
- Measurements of $\Pi(p, L)$:

$$\Pi(p, L) = \frac{\sum_{i=1}^{n} \pi_i}{n}, \tag{11}$$

where

$$\pi_i = \begin{cases} 1 & \text{if percolating} \\ 0 & \text{else.} \end{cases} \tag{12}$$

- Measurements of $P(p, L)$:
  $P(p, L) = \frac{\sum_{i=1}^{n} M_{S,i}}{nL^2}$
- Algorithm for finding spanning clusters

```python
def spanning_cluster_density(m, single_perc = False):
    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
            if single_perc:
                break

    return density
```

- Algorithm for estimating Π and P:

```python
def estimate_Pi_P(L_arr, p_arr, n_samples):
    for L in L_arr:
        Pi_arr = np.zeros(len(p_arr))
        density_arr = np.zeros(len(p_arr))
        for p, prob in enumerate(p_arr):
            for i in range(n_samples):
                z = np.random.random((L,L))
                m = z<prob
```

```
                density = spanning_cluster_density(m)
                density_arr[p] += density
                if density > 0:
                    Pi_arr[p] += 1

        Pi_arr /= n_samples
        density_arr /= n_samples
    return Pi_arr, density_arr
```

- Results for varying system sizes with $M = 1000$ samples, figure 5.1

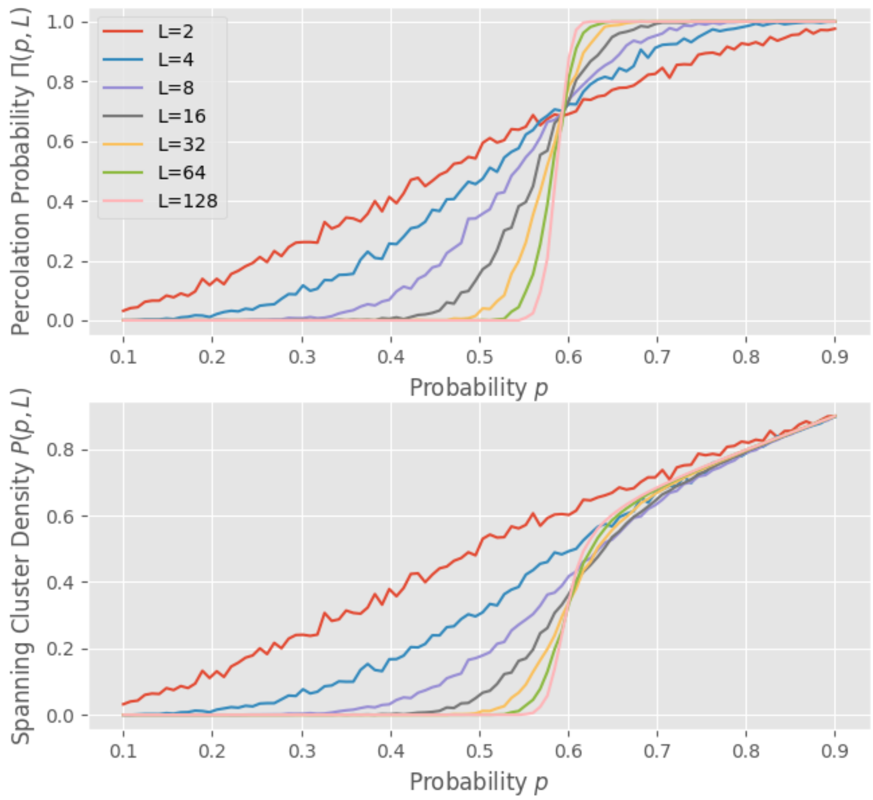


FIGURE 5.1. Spanning cluster density and percolation probability for different system sizes, $n = 1000$ samples.

- Π Linear for small systems, approaches step function for large systems.

- Expected behaviour for an infinite system:

$$\Pi(p, \infty) = \begin{cases} 0 & p < p_c \\ 1 & p \geq p_c \end{cases} \tag{13}$$

- Similar behaviour for P below the percolation threshold.

- Linear behaviour for $p > p_c$.

- Behaviour of $P(p_c, L)$ as the system size is increased.

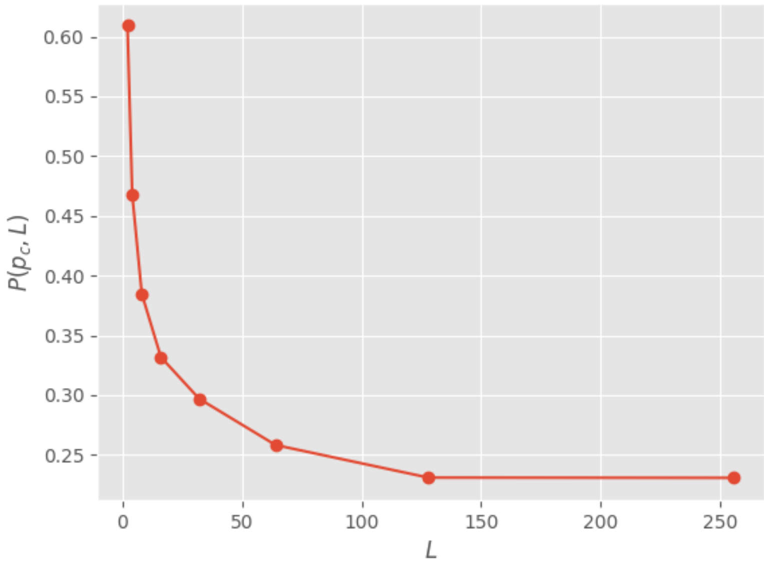


FIGURE 5.2. Spanning cluster density for different system sizes at $p = p_c$.

6. Topic 14: The cluster number density

Introduce the cluster number density and its applications: Definition, measurement, scaling and data-collapse.

- Definition of $n(s,p)$
- $P(\text{site is part of cluster of size } s) = sn(s,p)$
- Characterizes the size of clusters, basis for scaling theory in percolation
- Can be studied exactly in 1d, restricted to numerical estimates in 2d: $\overline{n(s,p)} = \frac{N_s}{ML^2}$.
- Measured by Monte Carlo simulations
- Logarithmic binning
- Effectively measures $\overline{n(s,p)}\Delta s_i$

```python
def log_bin(z, n_bins=10, base=10, log_max=None):
    if log_max is None:
        log_max = np.ceil(np.max(np.log(z))/np.log(base))
    log_bins = np.logspace(0, log_max, n_bins, base=base)
    log_hist, _ = np.histogram(z, bins=log_bins)
    dz = np.diff(log_bins)
    bin_mid = 0.5*(log_bins[1:] + log_bins[:-1])
    log_hist_normed = log_hist / dz

    return bin_mid, log_hist_normed
```

- Algorithm for measuring $n(s,p)$

```python
def cluster_number_density(L, p, n_samples, n_bins, logbase=10, log_max=None,
    remove_zeros=False):
    areas = []
    for i in tqdm(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, base=logbase, log_max=log_max)
    nsp = N_s/(n_samples*L**2)
    if remove_zeros:
```

```
        idx = np.where(nsp > 1e-15)[0]
        return s[idx], nsp[idx]
    else:
        return s, nsp
```

- $n(s,p)$ as a function of s in figure 6.1

- Can propose the following scaling form: $n(s,p) \propto s^{-\tau}$

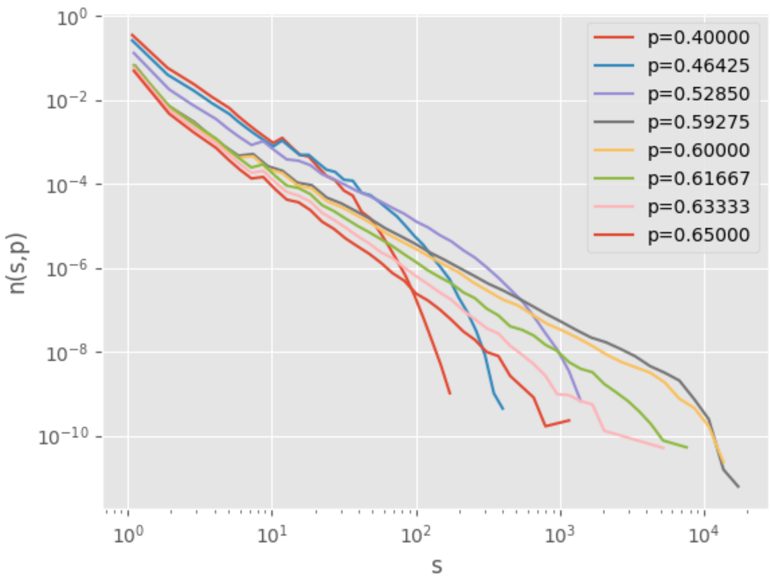


FIGURE 6.1. Plot of $n(s,p)$ for varying p in a 2d system of size $L = 200$.

- Estimating τ: $\log n(s,p) = -\tau \log s$

- Result in figure 6.2

- Estimated value: $\tau = 1.957$, "true" value: $\tau = 187/91 \approx 2.055$

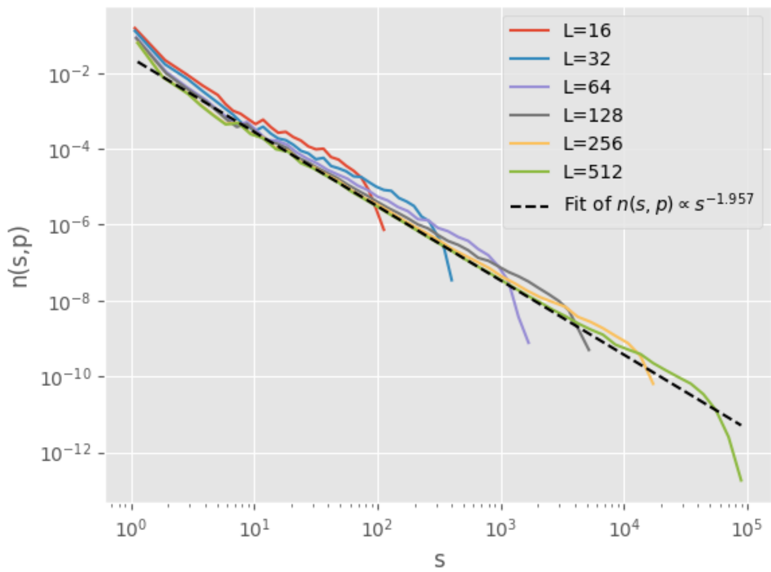


FIGURE 6.2. Cluster number density for varying system sizes at $p = p_c$. We also show the fitted exponent τ of the power law behaviour.

- Can also find s_ξ from these simulations

- Use result from 1d percolation to develop scaling theory of $n(s, p)$: $s_\xi \propto |p - p_c|^{-1/\sigma}$

- Propose that $n(s, p)$ has the form:

$$n(s, p) = s^{-\tau} F(s/s_\xi) = s^{-\tau} F\left((p - p_c)^{1/\sigma} s\right), \tag{14}$$

- Define s_ξ to be at the point $\frac{n(s,p)}{n(s,p_c)} = F(s/s_\xi) = 0.5$

- Algorithm for for finding s_ξ:

```
def estimate_s_xi(L, n_samples, n_bins):
    pc = 0.59275
    eps = 1e-14
    p_vals = [0.45, 0.50, 0.54, 0.57, 0.58]
    logbase = 1.3
    remove_zeros = False
    s, nc = cluster_number_density(L, pc, n_samples, n_bins, logbase,
    remove_zeros=remove_zeros)
    log_max = np.log(s[-1])/np.log(logbase)
    nonzero = np.where(nc > eps)[0]
    sxi_vals = []
    F_vals = []
```

```
    p_array = []
    for i, p in tqdm(enumerate(p_vals)):
        s, n = cluster_number_density(L, p, n_samples, n_bins, logbase, log_max, remove_zeros)
        nonzero = np.where(n > eps)[0]

        temp_sxi = []
        temp_n = []
        for i in range(len(n)):
            if n[i] > eps and nc[i] > eps:
                if n[i]/nc[i] <= 0.5:
                    temp_sxi.append(s[i])
                    temp_n.append(n[i])


        if len(temp_n)> 0:
            idx = np.argmax(temp_n)
            F = temp_n[idx]
            sxi = temp_sxi[idx]
            sxi_vals.append(sxi)
            F_vals.append(F)
            p_array.append(p)

    return sxi_vals, p_array
```

- Perform line fit to estimate σ:
  $\log s_\xi = -\frac{1}{\sigma} \log |p - p_c|$

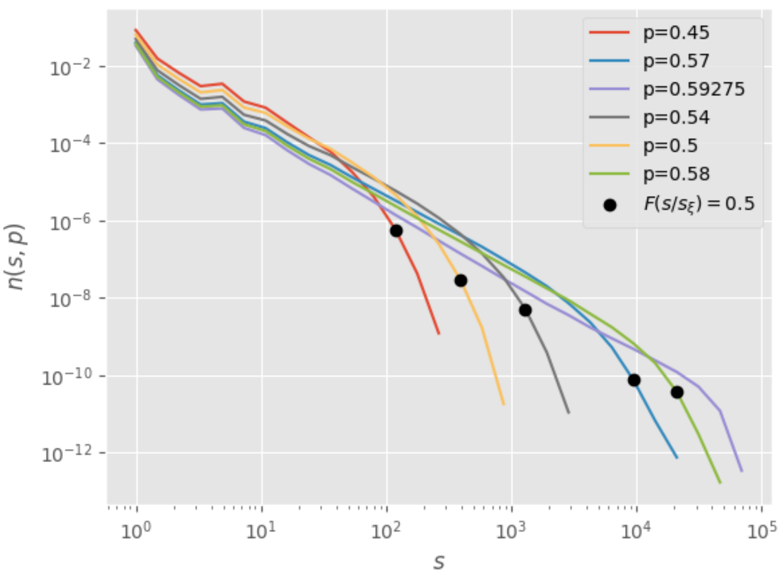


FIGURE 6.3. Cluster number density of a 2d system of size $L = 512$ for values of p approaching p_c.

- Our results: $\sigma = 0.462$, "true" value $\sigma = 36/91 \approx 0.395$

- Divergent behaviour of s_ξ shown in figure 6.4

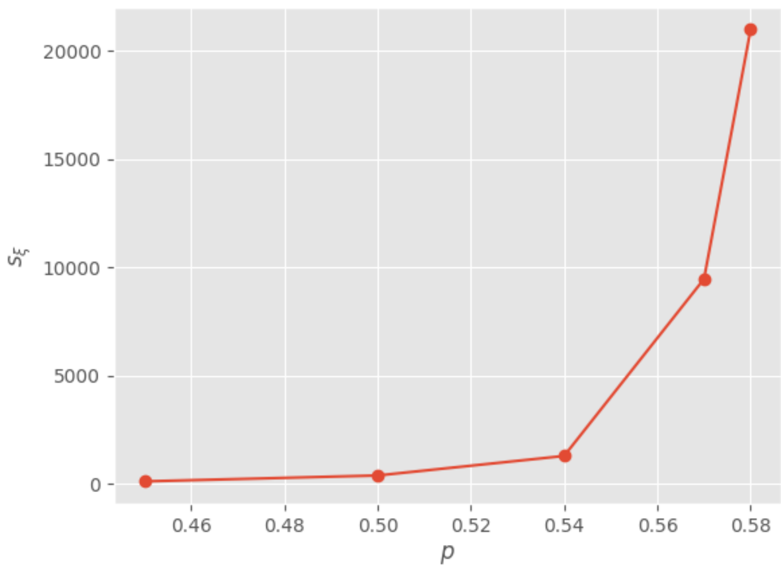


FIGURE 6.4. s_ξ as a function of p for a system of size $L = 512$.

- Data collapse: plot $s^\tau n(s,p)$ as a function of $s|p-p_c|^{1/\sigma}$, figure 6.5

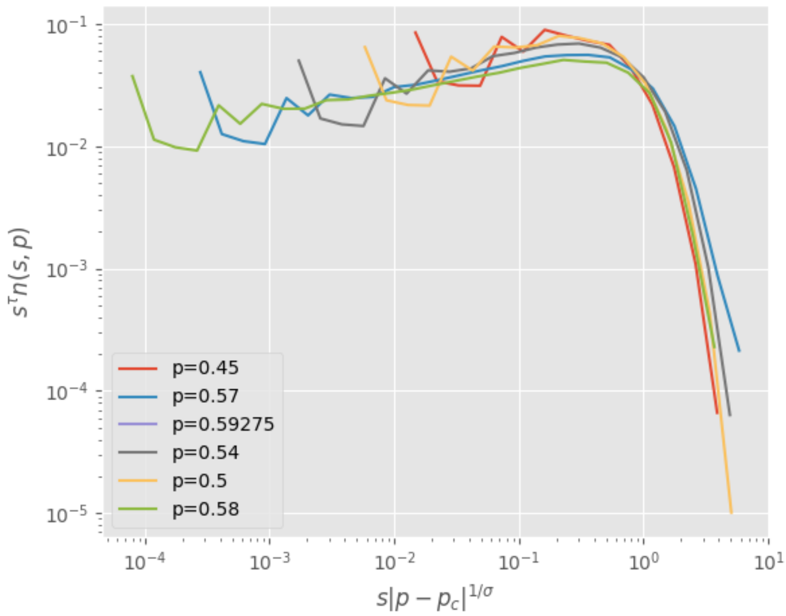


FIGURE 6.5. Data-collapse of $n(s,p)$.

7. Topic 15: Finite size scaling of $\Pi(p, L)$

Discuss the behavior of $\Pi(p, L)$ in a system with a finite system size L. How can we use this to find the scaling exponent ν, and the percolation threshold, p_c?

- Definition of $\Pi(p, L)$

- Expectations to it's behaviour

- Can be studied exactly in 1d and small 2d systems

- Restricted to numerical estimates in higher dimensions

- Use this as basis for a scaling theory
- Measurements by Monte Carlo simulations

$$\Pi(p, L) = \frac{\sum_{i=1}^{n} \pi_i}{n}, \tag{15}$$

where

$$\pi_i = \begin{cases} 1 & \text{if percolating} \\ 0 & \text{else.} \end{cases} \tag{16}$$

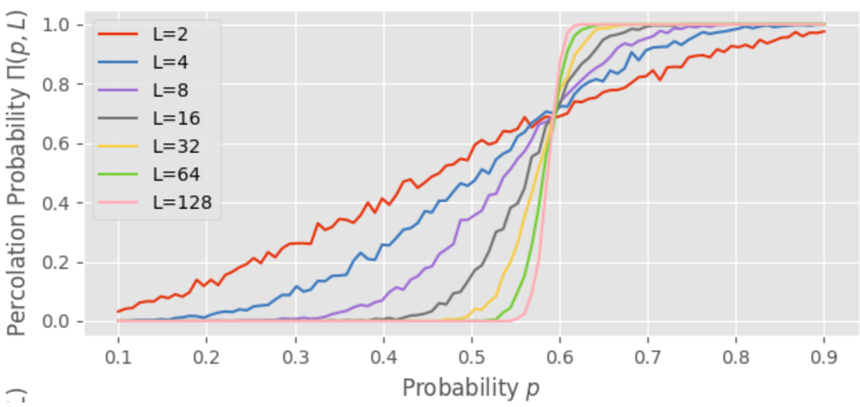
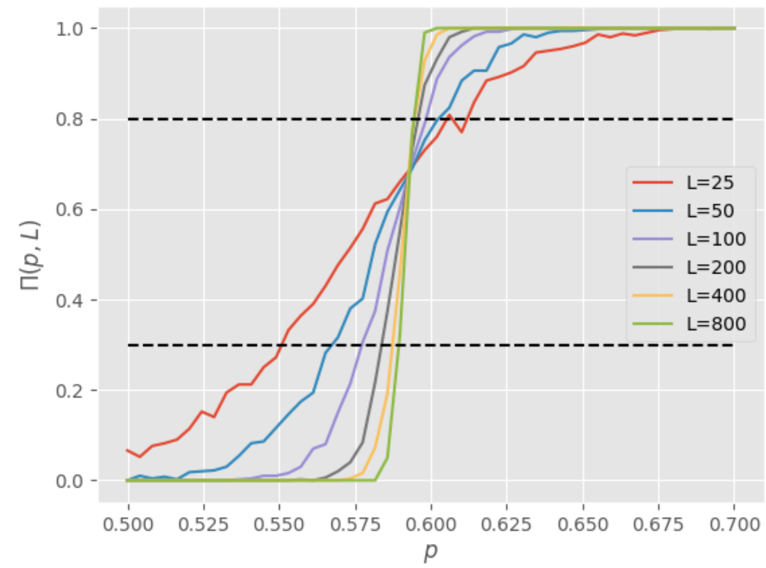


Figure 7.1. Percolation probability as a function of p for varying L.

- Behaviour of the different system sizes

- Estimating the percolation threshold



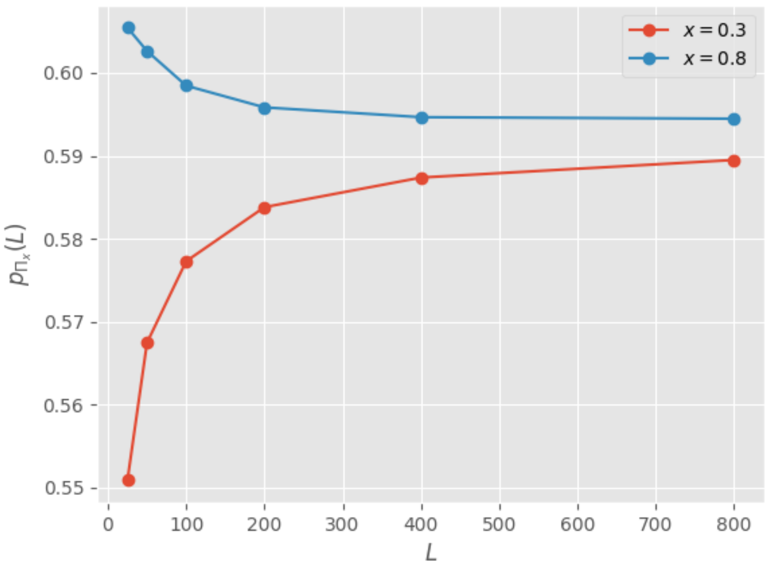


FIGURE 7.2. Plot of $\Pi(p, L)$ and p_{Π_x} at $x = 0.3, 0.5$ for varying system sizes.

- For an infinite system:

$$\Pi(p, \infty) = \begin{cases} 0 & p < p_c \\ 1 & p \geq p_c, \end{cases} \tag{17}$$

- Not a viable strategy for estimating p_c

- Instead develop a scaling theory
- From above results: Π is dependent on both ξ and L

- Proposed scaling form: $\Pi(p, L) = \xi f\left(\frac{L}{\xi}\right)$

- Use $\xi = \xi_0 |p - p_c|^{-\nu}$:

- $\Pi(p, L) = f(L\xi_0 |p - p_c|^{\nu}) = f(\xi_0 (L^{1/\nu}(p - p_c))^{\nu})$

- Introduce $\phi(u) = f(\xi_0 u^{1/\nu})$

- Finite size scaling ansatz: $\Pi(p, L) = \phi(L^{1/\nu}(p - p_c))$
- Goal: use this result to estimate p_c and ν

- Problem if both are unknown

- From above results and figure:

$$x = \Pi(p_x, L) = \phi(L^{1/\nu}(p_x(L) - p_c)) \tag{18}$$
$$\phi^{-1}(x) = (p_x - p_c)L^{1/\nu} = C_x. \tag{19}$$

- Subtract equation from itself:

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

- Estimate ν by line fit:

- $\log dp = \log(C_{x_1} - C_{x_2}) - \frac{1}{\nu}\log(L)$
- Estimated: $\nu = 1.405$, "true" value $\nu = 4/3 \approx 1.333$

- Can now use this to estimate p_c:

- $p_x(L) = p_c + C_x L^{-1/\nu}$

- Results in figure 7.3

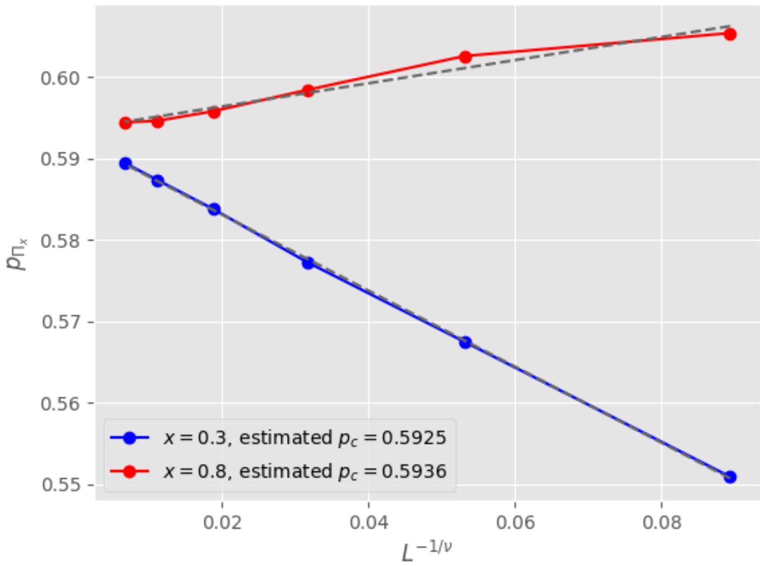


FIGURE 7.3. Plot of p_x as a function of $L^{-1/\nu}$ for $x = 0.3$ and $x = 0.8$. The y-intercept is the estimated value for p_c.

- Shape of the scaling function $\phi(x)$:

- Plot $\Pi(p, L)$ as a function of $(p - p_c)L^{-1/\nu}$

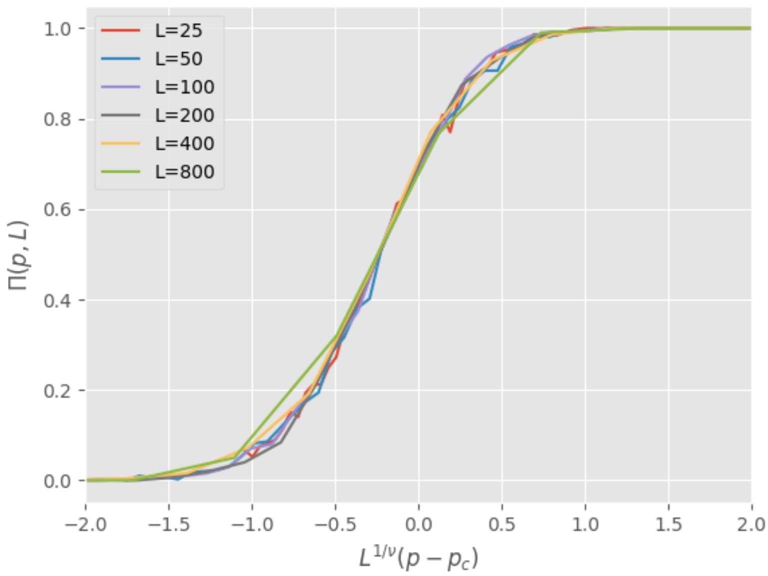


Figure 7.4. Data-collapse of $\Pi(p, L)$.

8. Topic 16: Effective percolation threshold

Define and discuss the effective percolation threshold in a system with a finite system size L. How can we measure it, and what consequences does it have that the effective percolation threshold is different from the actual percolation threshold?

- System of finite size L will have a probability p which on average gives rise to a spanning cluster
- Dependent on system size, $p_x(L)$
- Approaches p_c as $L \to \infty$
- Definition of effective percolation threshold: $\Pi(p_x, L) = x$
- Plot of $\Pi(p, L)$, figure 8.1

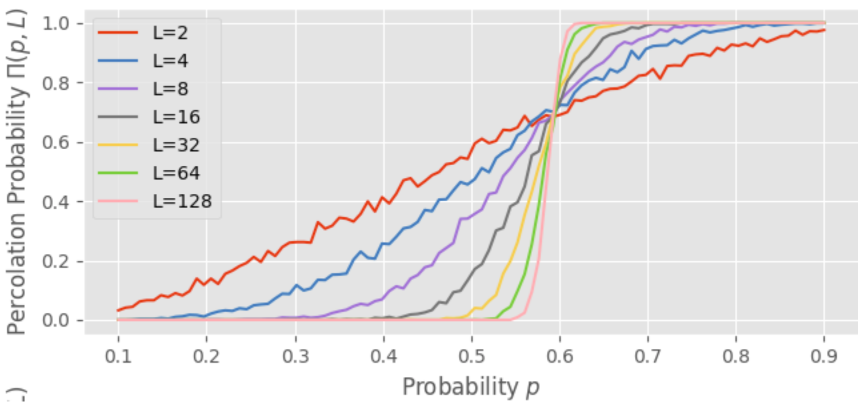


Figure 8.1. Percolation probability as a function of p for different L.

- Simplest approach: define p_x where $\Pi(p, L) = 1/2$.
- Can measure $p_x(L)$ by interpolation:

```
x_vals = [0.3, 0.8]
L_vals = [25, 50, 100, 200, 400, 800]
for j, x in enumerate(x_vals):
    for i, L in enumerate(L_vals):
        Pi = data[i,:] #data contains Pi_p_L loaded from file
        idx = np.argmax(Pi > x)
        pc = p[idx-1] + (x-Pi[idx-1])*(p[idx] - p[idx-1])/(Pi[idx] - Pi[idx-1])
        p_pi_vals[j,i] = pc
```

- Results from calculations of Π with $M = 500$ samples, figure 8.2

- Convergence towards a finite value

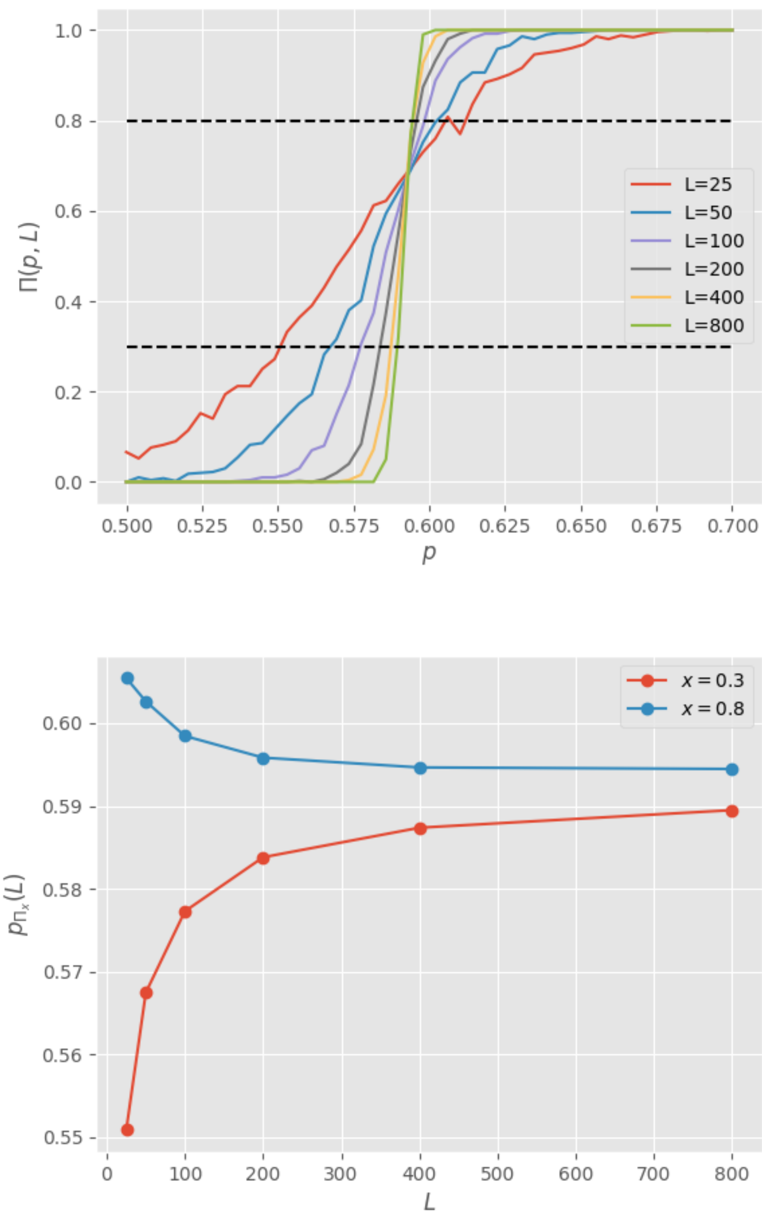


FIGURE 8.2. Plot of $\Pi(p, L)$ and p_{Π_x} at $x = 0.3, 0.5$ for varying system sizes.

- Consequences for $p_x \neq p_c$:

- Find scaling of Π and ξ, and predict p_c
- From scaling theory of $\Pi(p, L)$:

$$\Pi(p, L) = \xi f(L/\xi) = f\left(\xi_0 \left(L^{1/\nu}(p - p_c)\right)^{\nu}\right) \tag{20}$$

$$= \phi\left(L^{1/\nu}(p - p_c)\right), \tag{21}$$

where

$$\phi(u) = f\left(\xi_0 u^{1/\nu}\right). \tag{22}$$

- Estimating the scaling of ξ:
- Apply $\phi^{-1}(x)$ and use $\Pi(p_x, L) = x$:
- $(p_x(L) - p_c)L^{1/\nu} = \phi^{-1}(x) = C_x$
- Fit line to our data: $\log(p_x(L) - p_c) = \log(C_x) - \frac{1}{\nu}\log(L)$
- Requires that p_c is known. Can instead use:
- $dp = \log(C_{x_1} - C_{x_2}) - \frac{1}{\nu}\log(L)$
- Estimated: $\nu = 1.405$, "true" value $\nu = 4/3 \approx 1.333$.
- Can now estimate p_c if we wish, results in figure 8.3

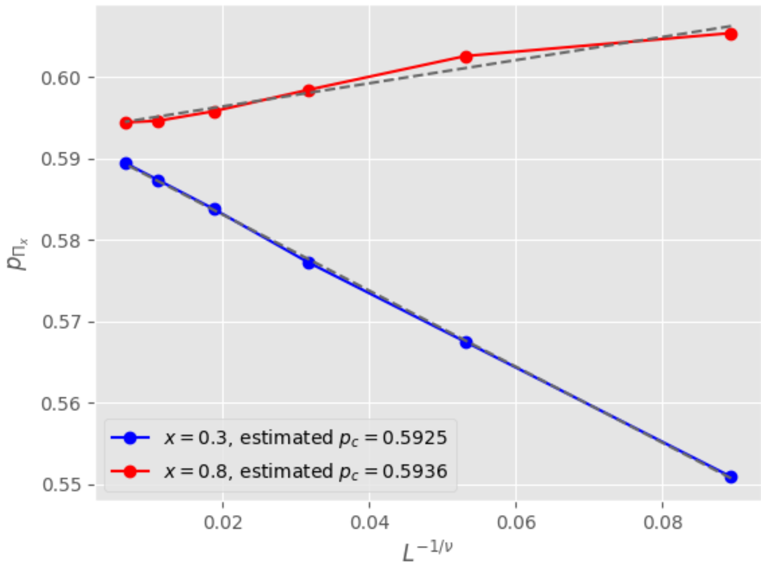


FIGURE 8.3. Plot of the effective percolation threshold for $x = 0.3$ and $x = 0.8$. The y-intercept is the estimated value for p_c.

9. Topic 17: Subsets of the spanning cluster

Introduce and discuss the scaling of subsets of the spanning cluster. How can we measure the singly-connected bonds, and how does it scale?

- Definition of the subsets of the spanning cluster:
- Singly connected bonds
- Backbone
- Dangling ends
- visualization of the subsets in figure 9.1

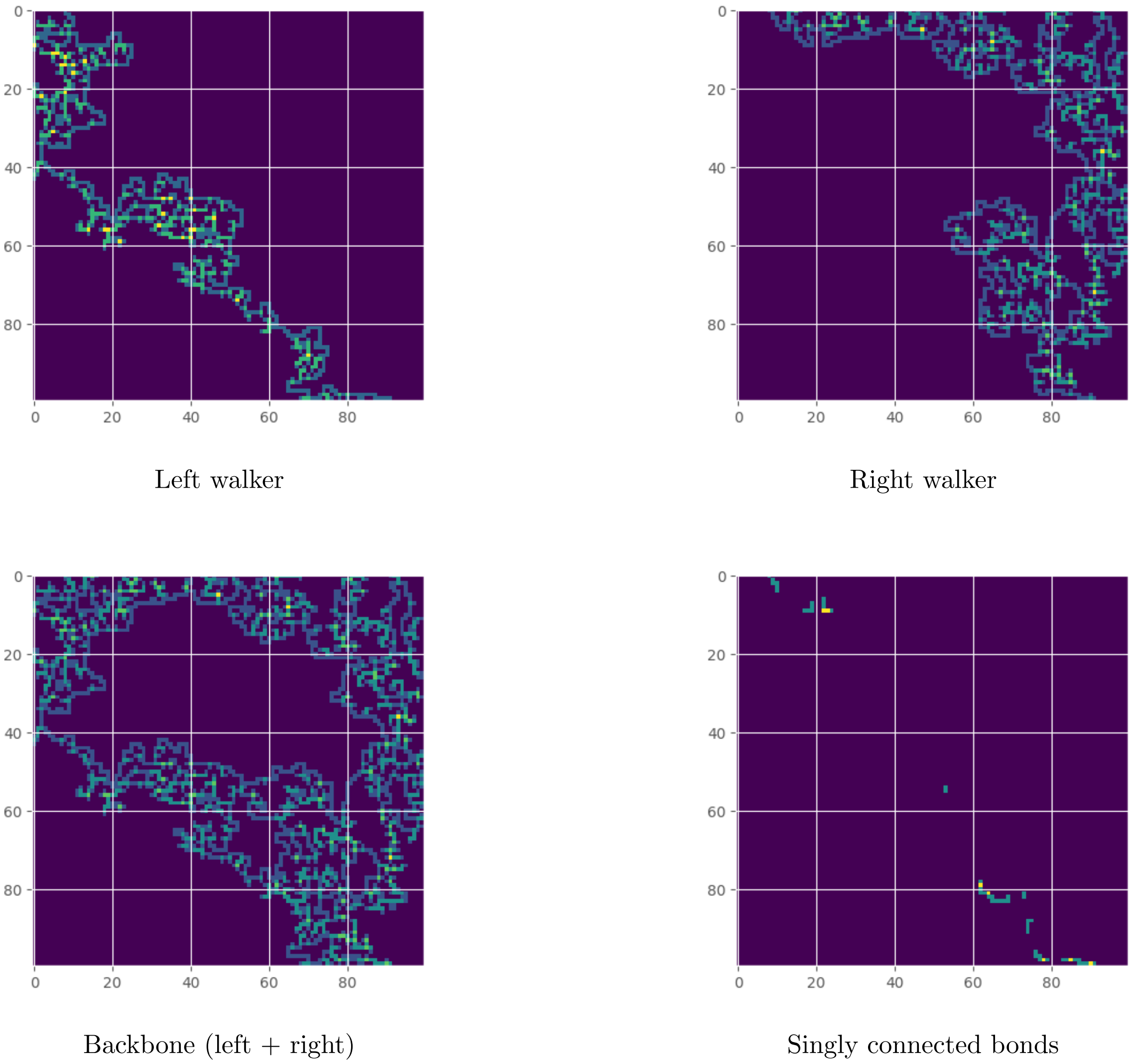


Figure 9.1. Visualization of the paths of the right and left walkers.

- Scaling of the subset geometries
- Propose similar scaling form as $M \propto L^D$

$$M_{SC} \propto L^{D_{SC}} \tag{23}$$
$$M_{min} \propto L^{D_{min}} \tag{24}$$
$$M_{max} \propto L^{D_{max}} \tag{25}$$
$$M_{BB} \propto L^{D_{BB}} \tag{26}$$
$$M_{DE} \propto L^{D_{DE}}. \tag{27}$$

- Hierarchy of the exponents:

$$D_{SC} \leq 1 \leq D_{min} \leq D_{max} \leq D_{BB} \leq D \leq d. \tag{28}$$

- Scaling of dangling ends:
- $M_{DE} = M - M_{BB} \propto L^D - L_{BB}^D = L^{D-D_{BB}}$
- For an infinite system:
- $\frac{M_{DE}}{M} \propto 1 - \frac{L^{D_{BB}}}{L^D} = 1 - L^{D_{BB}-D}$
- Scaling of the singly connected bonds
- Use Monte Carlo simulation
- Measure M_{SC} for $L = 2^k$ for $k \in [4, 10]$.
- Linear fit of $\log M_{SC} \propto D_{SC} \log L$,. Figure 9.2
- Estimated: $D_{SC} = 0.7822$, "true" value: 0.75

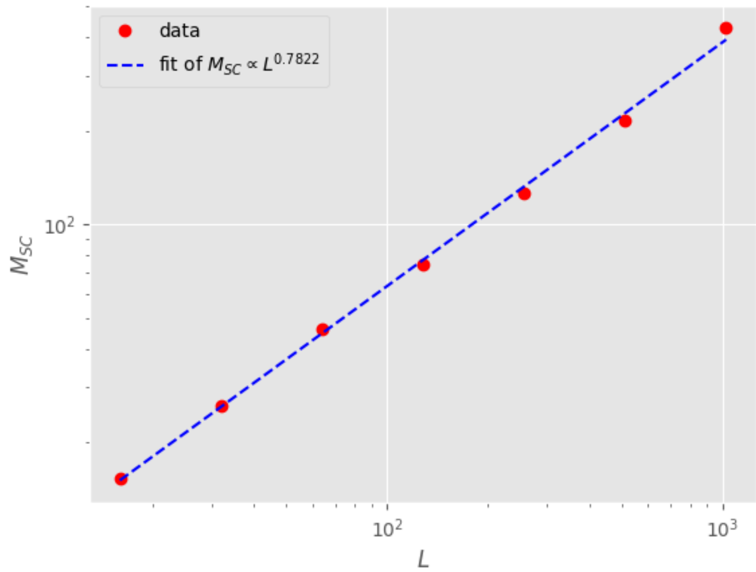


FIGURE 9.2. Mass of the singly connected bonds as a function of system size.

- Density of the singly connected bonds:
- $P_{SC} = \frac{M_{SC}}{L^d} \propto L^{D_{SC}-d}$
- Measure as a function of $(p - p_c)$
- Plot in figure 9.3

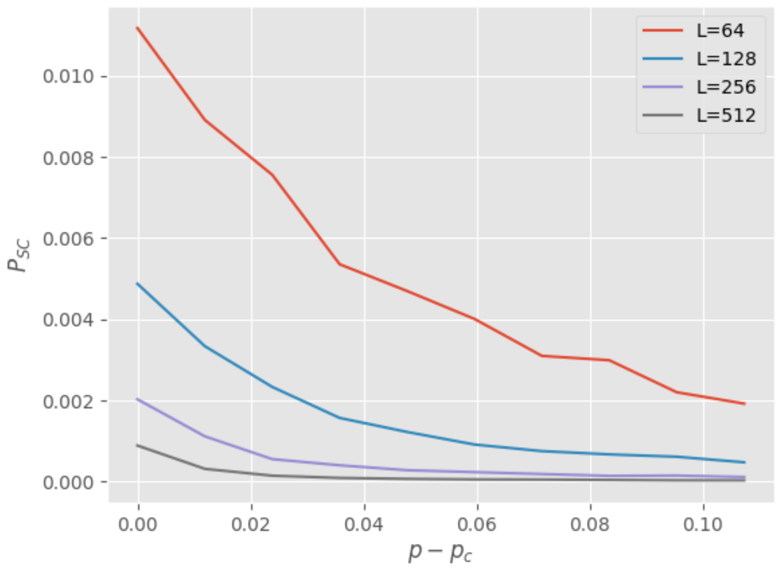


FIGURE 9.3. Density of the singly connected bonds as a function of $p - p_c$ for various L.

- Power-law behaviour of the density
- Propose $P_{SC} \propto (p - p_c)^{-x}$
- Fit to the data: $x = 1.06$. Plot in figure 9.4

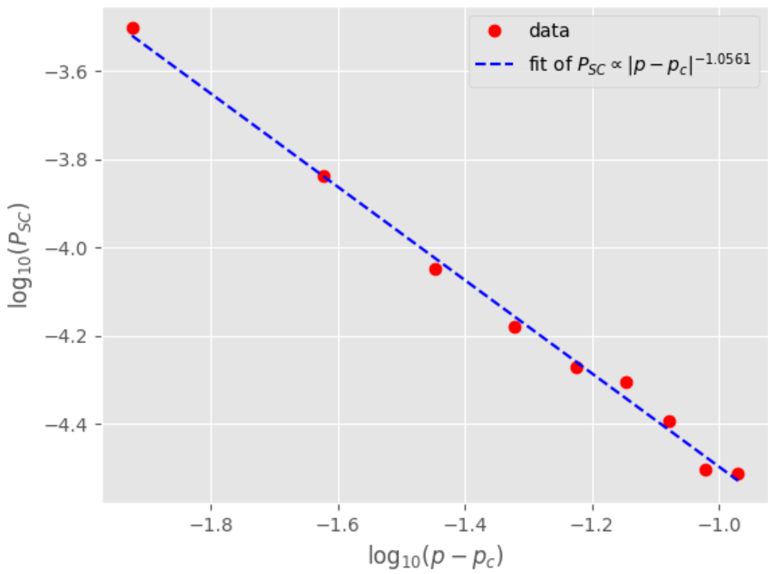


FIGURE 9.4. Loglog plot of P_{SC} as a function of $(p - p_c)$ for a sysstem of size $L = 512$