PH415 Assignment 3
Percolation

S Venkat Bharadwaj

October 27, 2023

Problem

Consider a site percolation on 2D lattice for L equal to 40-160 in steps of 20. Populate the lattice with varying occupation probability from 0.5 to 0.7 in steps of 0.002. Apply Hoshen-Kopelman algorithm to identify different clusters and their sizes (s). Calculate the Order Parameter ($P_\infty(p, L)$), Average Cluster Size Distribution ($\chi(p, L)$) and Binder Cumulant (U(p,L)). From these, Evaluvate the following:

i Plot U(p,L) against p for various L and determine percolation threshold (p_c) for a system of infinite size. Determine the value of ν.

ii Plot $P_\infty(p, L)$ against p for various L and determine the value of β/ν. Verify the scaling form.

iii Plot $\chi(p, L)$ against p for various L and determine the value of γ/ν. Verify the scaling form.

iv Verify the Scaling relation between β/ν, γ/ν and the space dimension, d.

General Algorithm

- Step 1: Generate a LxL lattice. (Initially, L=40 and p=0.5)
- Step 2: Generate a random number,r, LxL times. If r<p, populate the corresponding lattice site.
- Step 3: Implement the Hoshen-Kopelman Algorithm.
- Step 4: Compare the 1st and last row of the HK matrix to see of same 'k' values are repeating. If Yes, the system is percolating and the mass corresponding to the 'k' values are infinite cluster size , s_∞.
- Step 5: From the mass array, ignore the negative values and infinite cluster. By counting the frequency the remaining values, we can get size of cluster, s and number of clusters of size s, n_s values.
- Step 6: Using the s and n_s values calculate $P_\infty(p, L)$, $\chi(p, l)$, $P_\infty(p, L)^4$, $P_\infty(p, L)^2$.

$$P_\infty(p, L) = p - \sum_s \frac{s * n_s}{L^2} \quad or \quad P_\infty(p, L) = \sum \frac{S_\infty}{L^2}$$

$$\chi(p, L) = \sum_s \frac{s^2 * n_s}{L^2}$$

- Step 7: Increase p by 0.002 and repeat from Step 2, until p=0.7. Then go to Step 8.
- Step 8: Increase L by 20 and repeat from Step 1, until L=160. The go to Step 9.
- Step 9: Repeat the above algorithm N=10000 times. Find the average values of $P_\infty(p, L)$, $\chi(p, l)$, $P_\infty(p, L)^4$, $P_\infty(p, L)^2$ for better results.
- Step 10: From the average value of $P_\infty(p, L)^4$, $P_\infty(p, L)^2$, Calculate U(p,L) for all p,L.

$$U(p, L) = 1 - \frac{< P_\infty^4 >}{3 < P_\infty^2 >^2}$$

Hoshep Kopelman Algorithm

- Start from the top left of the lattice. Here, we traverse the lattice left to right and top to bottom. Create an array named mass.
- If lattice is occupied, there are 3 possible situations for the top and left neighbour.
  a) Both are unoccupied. In this case, label the cluster as k(starting from k=1). Increase mass(k) by 1 and increase k by 1.
  b) One of them is occupied or both are occupied belonging to same cluster. In this case, label the cluster same as the occupied neighbour. Increase the mass corresponding to the cluster by 1.
  c) Both are occupied belonging to different cluster. In this case, the clusters need to united into 1. The mass corresponding to the cluster with larger label(k1) is added to the mass corresponding to the smaller label(k2). M(k1) is increased by 1. M(k2) is set to -k1 to denote the change of cluster.

Code

Implementing of Percolation (MATLAB)

```
L = 40:20:160;
P = 0.5:0.002:0.7;
N=10000;
pinf = zeros(length(L),length(P));
csd = zeros(length(L),length(P));
bc = zeros(length(L),length(P));
a = zeros(length(L),length(P));
b = zeros(length(L),length(P));

for n = 1:N
    for l = 1:length(L)
        for p = 1:length(P)
            lat = zeros(L(l),L(l));
            %count=0;
            for i = 1:L(l)
                for j = 1:L(l)
                    r = rand;
                    if r<P(p)
                        lat(i,j) = 1;
                        %count=count+1;
                    end
                end
            end
            %disp(count/(L(l)*L(l)))
            %disp(lat)
            hk = zeros(L(l),L(l));
            mass = [0];
            k=1;
            for i = 1:L(l)
                for j = 1:L(l)
                    if i==1
                        if j==1
                            if lat(i,j)
                                hk(i,j) = k;
                                k=k+1;
                                mass = [mass,0];
                                mass(hk(i,j)) = mass(hk(i,j))+1;
                                continue;
                            else
                                continue;
                            end
                        else
                            if lat(i,j)
                                if lat(i,j-1)
                                    hk(i,j) = hk(i,j-1);
```

```
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                            continue;
                        else
                            hk(i,j) = k;
                            k=k+1;
                            mass = [mass,0];
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                            continue;
                        end
                    end
                end
            end
            if j==1
                if lat(i,j)
                    if lat(i-1,j)
                        tmp = mass(hk(i-1,j));
                        tmp1= hk(i-1,j);
                        while tmp<0
                            tmp1 = tmp;
                            tmp = mass(abs(tmp1));
                        end
                        hk(i,j) = abs(tmp1);
                        mass(hk(i,j)) = mass(hk(i,j))+1;
                        continue;
                    else
                        hk(i,j) = k;
                        k=k+1;
                        mass = [mass,0];
                        mass(hk(i,j)) = mass(hk(i,j))+1;
                        continue;
                    end
                end
            end
            if lat(i,j)
                if lat(i-1,j)
                    if lat(i,j-1)
                        if hk(i,j-1) == hk (i-1,j)
                            hk(i,j) = hk(i-1,j);
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                            continue;
                        end
                        tmp = mass(hk(i-1,j));
                        tmp1= hk(i-1,j);
                        while tmp<0
                            tmp1 = tmp;
                            tmp = mass(abs(tmp1));
                        end
                        tmp = mass(hk(i,j-1));
                        tmp2= hk(i,j-1);
                        while tmp<0
                            tmp2=tmp;
                            tmp = mass(abs(tmp2));
                        end
                        if abs(tmp1)==abs(tmp2)
                            hk(i,j) = abs(tmp1);
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                        else
                            hk(i,j) = min(abs(tmp1),abs(tmp2));
                            mass(hk(i,j)) = mass(abs(tmp1))+mass(abs(tmp2))+1;
                            mass(max(abs(tmp1),abs(tmp2))) = -1*hk(i,j);
                        end
                    else
                        tmp = mass(hk(i-1,j));
                        tmp1=hk(i-1,j);
                        while tmp<0
                            tmp1 = tmp;
                            tmp = mass(abs(tmp1));
                        end
                        hk(i,j) = abs(tmp1);
                        mass(hk(i,j)) = mass(hk(i,j))+1;
                    end
```

```
                        elseif lat(i,j-1)
                            tmp = mass(hk(i,j-1));
                            tmp1= hk(i,j-1);
                            while tmp<0
                                tmp1 = tmp;
                                tmp = mass(abs(tmp1));
                            end
                            hk(i,j) = abs(tmp1);
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                        else
                            hk(i,j) = k;
                            k=k+1;
                            mass = [mass,0];
                            mass(hk(i,j)) = mass(hk(i,j))+1;
                        end
                    end
                end
            end
            %disp(hk)
            s = [];
            ns = [];
            for t = 1:length(mass)
                if mass(t)>0
                    if ismember(mass(t),s)
                        continue;
                    else
                        s = [s,mass(t)];
                        bla = 0;
                        for i = 1:length(mass)
                            if mass(i)==mass(t)
                                bla = bla+1;
                            end
                        end
                        ns = [ns,bla];
                    end
                end
            end
            %disp(s)
            %disp(ns)
            perc = [];
            for i = 1:L(l)
                if hk(1,i) ~=0
                    for j = 1:L(l)
                        if hk(L(l),j) == hk(1,i)
                            if ismember(hk(1,i),perc)
                                continue;
                            else
                                perc = [perc,hk(1,i)];
                            end
                        end
                    end
                end
            end
            %disp(perc)
            massinf = zeros(length(perc),1);
            tmp = 0;
            for i = 1:length(perc)
                massinf(i) = mass(perc(i));
                tmp = tmp+massinf(i)/(L(l)*L(l));
            end
            for i = 1:length(s)
                if ismember(s(i),massinf)
                    continue;
                else
                    %tmp = tmp-s(i)*ns(i)/(L(l)*L(l));
                    csd(l,p) = csd(l,p)+s(i)*s(i)*ns(i)/(L(l)*L(l));
                end
            end
            pinf(l,p) = tmp+pinf(l,p);
            a(l,p) = a(l,p)+(tmp)^4;
            b(l,p) = b(l,p)+(tmp)^2;
```

```
            end
        end
        disp(n)
end
pinf = pinf/N;
csd = csd/N;
a = a;
b = b;
for l = 1:length(L)
    for p = 1:length(P)
        bc(l,p) = (1-(a(l,p)/(3*b(l,p)^2)));
    end
end
%disp(pinf1)
figure(1);
title('Order Parameter (P_{inf}) vs Occupation Probability (p) for various Lattice
    size')
xlabel('p')
ylabel('P_{inf}')
hold on;
for i = 1:length(L)
    plot(P,pinf(i,:))
end
legend({'L = 40','L = 60','L = 80','L = 100','L = 120','L = 140','L = 160'})
hold off;

figure(2);
title('Average Cluster Size (Chi) vs Ocupation Probability (p) for various Lattice
    size')
xlabel('p')
ylabel('Chi')
hold on;
for i = 1:length(L)
    plot(P,csd(i,:))
end
legend({'L = 40','L = 60','L = 80','L = 100','L = 120','L = 140','L = 160'})
hold off;

figure(3);
hold on;
title('Binder Cumulant (U) vs Occupation Probability (p) for various Lattice size')
xlabel('p')
ylabel('U')
for i = 1:length(L)
    plot(P,bc(i,:))
end
legend({'L = 40','L = 60','L = 80','L = 100','L = 120','L = 140','L = 160'})
hold off;

writematrix(pinf, 'Pinf.csv')
writematrix(csd, 'csd.csv')
writematrix(bc, 'bc.csv')
```

Plots and Finding the Exponents(Python)

```
from cProfile import label
from operator import le
import numpy as np
import matplotlib.pyplot as plt
import math as m

L = [40,60,80,100,120,140,160]
P = np.arange(0.5,0.701,0.002)
x = np.zeros((len(L)))
y = np.zeros((len(L)))

Pinf = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment
     3\Pinf.csv",delimiter=",", dtype=float)
csd = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment
    3\csd.csv",delimiter=",", dtype=float)
bc = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment
```

```
    3\Bc.csv",delimiter=",", dtype=float)

pc = 0.596
Pinf_pc = Pinf[:,int((pc-0.5)/0.002)]
for i in range(0,len(L)):
    y[i] = m.log(Pinf_pc[i])
    x[i] = m.log(L[i])
plt.scatter(x,y)
z = np.polyfit(x,y, 1)
p = np.poly1d(z)
bn = -z[0]
plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
plt.title('Log-Log Scale Plot of P_inf at Percolation Threshold vs. Lattice Size')
plt.ylabel('log(P_inf(p=p_c,L))')
plt.xlabel('log(L)')
plt.legend()
plt.show()

chi_pc = csd[:,int((pc-0.5)/0.002)]
for i in range(0,len(L)):
    y[i] = m.log(chi_pc[i])
    x[i] = m.log(L[i])
plt.scatter(x,y)
z = np.polyfit(x,y, 1)
p = np.poly1d(z)
gn = z[0]
plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
plt.title('Log-Log Scale Plot of Chi at Percolation Threshold vs. Lattice Size')
plt.ylabel('log(Chi(p=p_c,L))')
plt.xlabel('log(L)')
plt.legend()
plt.show()

dU_pc = (bc[:,int((pc-0.5)/0.002)]-bc[:,int((pc-0.5)/0.002)-1])/0.002
for i in range(0,len(L)):
    y[i] = m.log(dU_pc[i])
    x[i] = m.log(L[i])
plt.scatter(x,y)
z = np.polyfit(x,y, 1)
p = np.poly1d(z)
nu = 1/z[0]
plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
plt.title('Log-Log Scale Plot of dU/dp at Percolation Threshold vs. Lattice Size')
plt.ylabel('log(dU(p,L)/dp)|p=p_c')
plt.xlabel('log(L)')
plt.legend()
plt.show()

for i in range(0,len(L)):
    x = np.zeros((len(P)))
    y = Pinf[i,:]
    for j in range(0,len(P)):
        x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
        y[j] = y[j]*m.pow(L[i],bn)

    plt.plot(x,y, label='L = '+str(L[i]))
plt.title('Rescaled Plot of P_inf(p,L) vs p for various L')
plt.xlabel(r"$z = (p-p_c)*L^{1/\nu}$")
plt.ylabel(r"$L^{\beta/\nu}P_\infty$")
plt.legend()
plt.show()

for i in range(0,len(L)):
    x = np.zeros((len(P)))
    y = csd[i,:]
    for j in range(0,len(P)):
        x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
        y[j] = y[j]*m.pow(L[i],-gn)

    plt.plot(x,y, label='L = '+str(L[i]))
plt.title('Rescaled Plot of Chi(p,L) vs p for various L')
```

```
plt.xlabel(r"$z = (p-p_c)*L^{1/\nu}$")
plt.ylabel(r"$L^{-\gamma/\nu}\chi$")
plt.legend()
plt.show()

for i in range(0,len(L)):
    x = np.zeros((len(P)))
    y = bc[i,:]
    for j in range(0,len(P)):
        x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
        y[j] = y[j]

    plt.plot(x,y, label='L = '+str(L[i]))
plt.title('Rescaled Plot of U(p,L) vs p for various L')
plt.xlabel(r"$z = (p-p_c)*L^{1/\nu}$")
plt.ylabel(r"$U(p,L)$")
plt.legend()
plt.show()
```

Results

Order Parameter (P_∞)

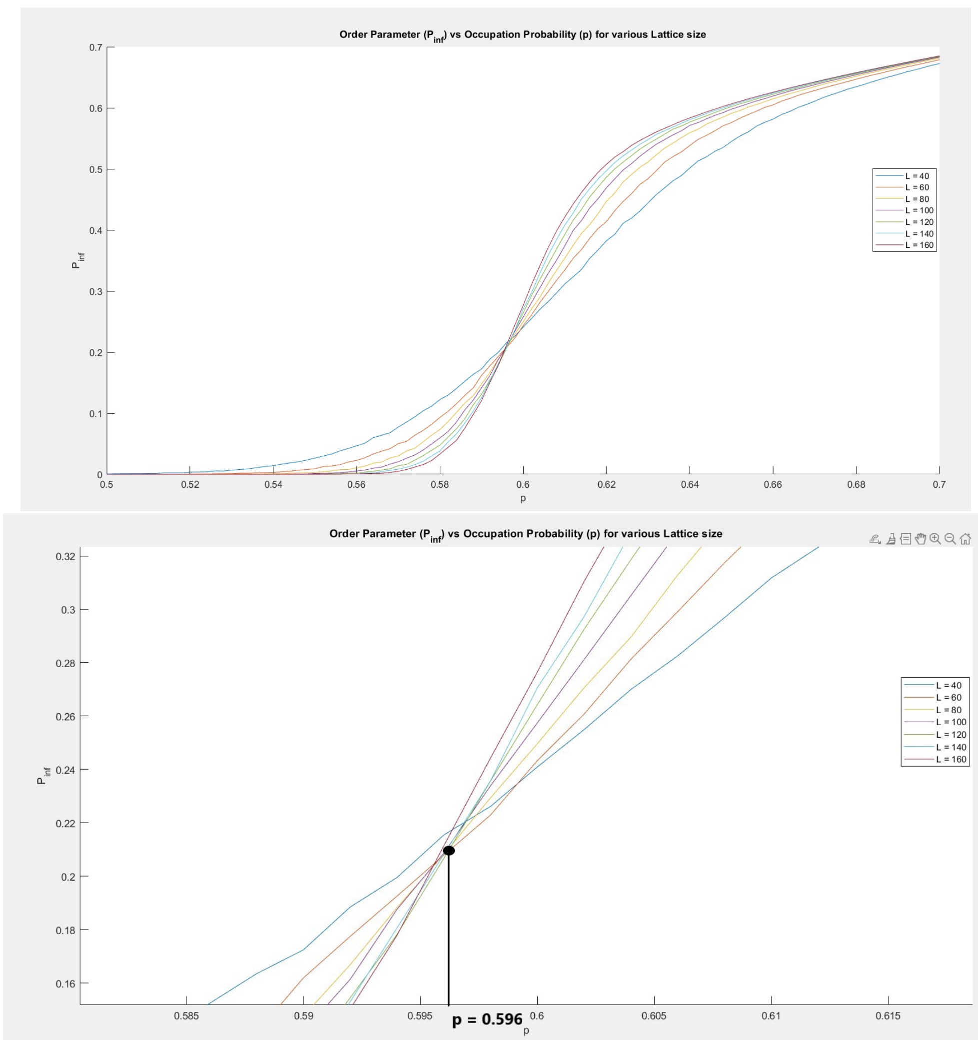

Figure 1: Order Parameter vs. Occupation Probability averaged over N=10000 iterations.

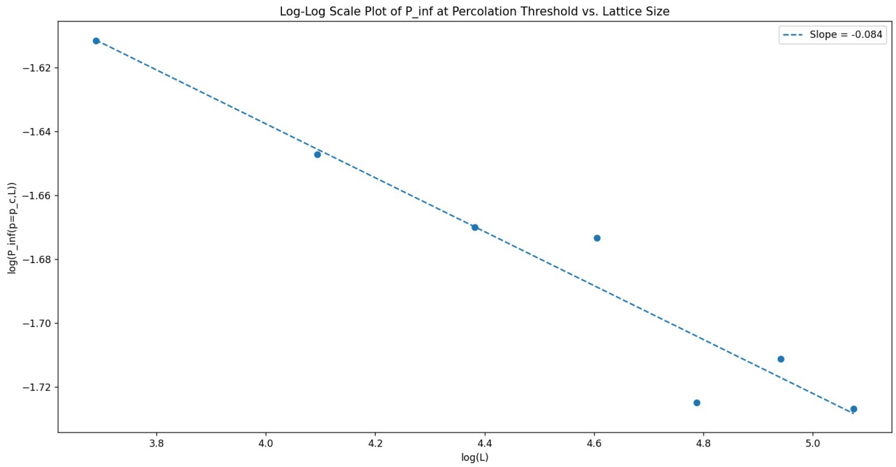

Figure 2: Order Parameter at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent β/ν

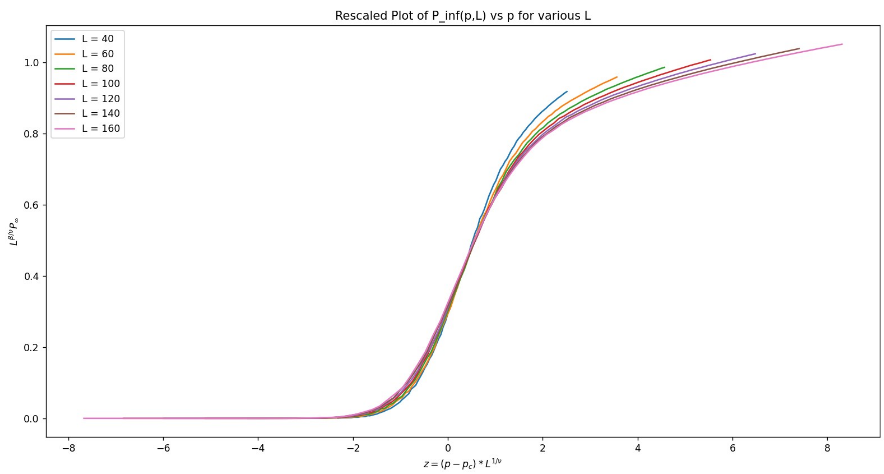

Figure 3: Order Parameter vs. Occupation Probability after rescaling using the exponents ν, β/ν.

Average Cluster Size (χ)

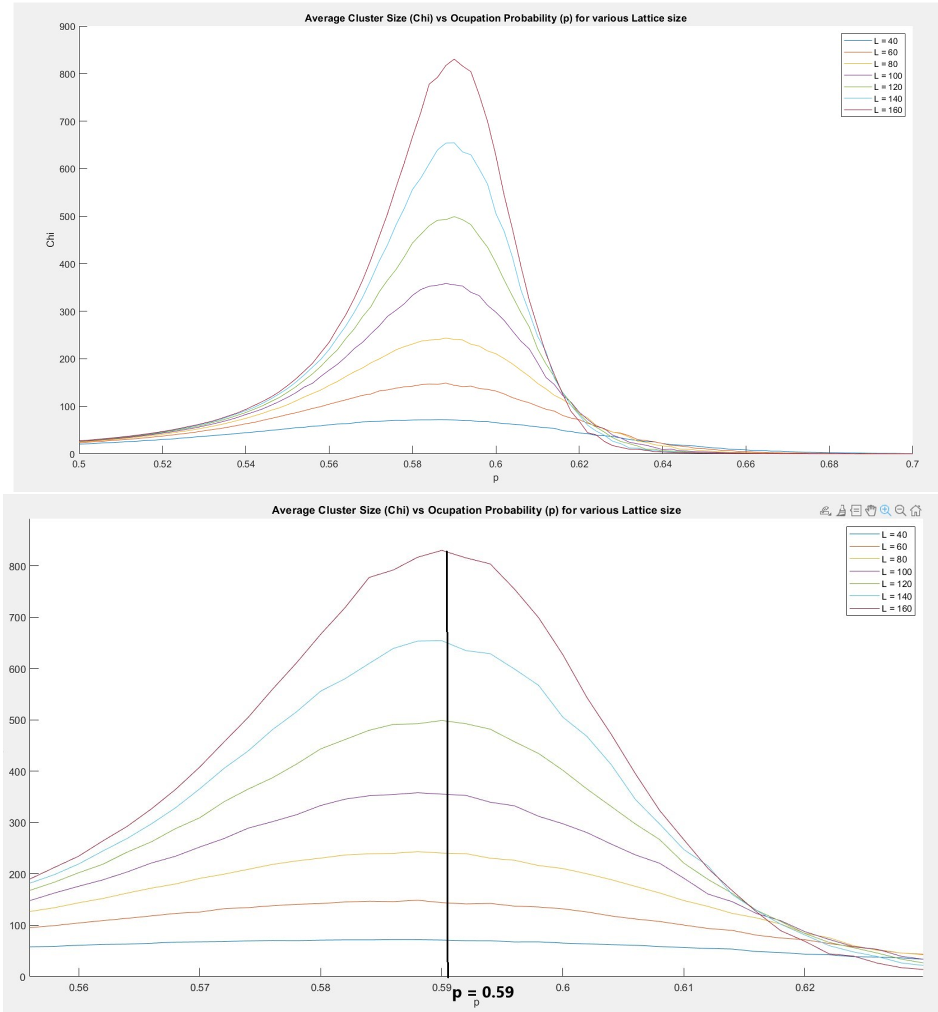

Figure 4: Average Cluster Size vs. Occupation Probability averaged over N=10000 iterations.

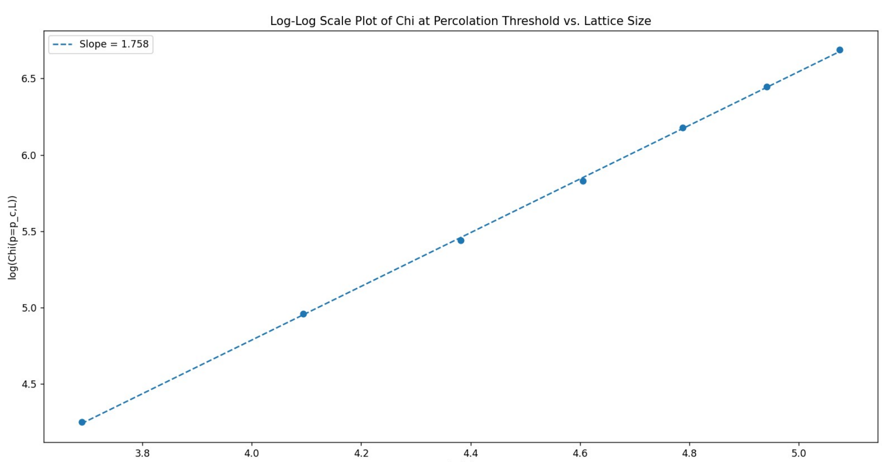

Figure 5: Average Cluster Size at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent γ/ν

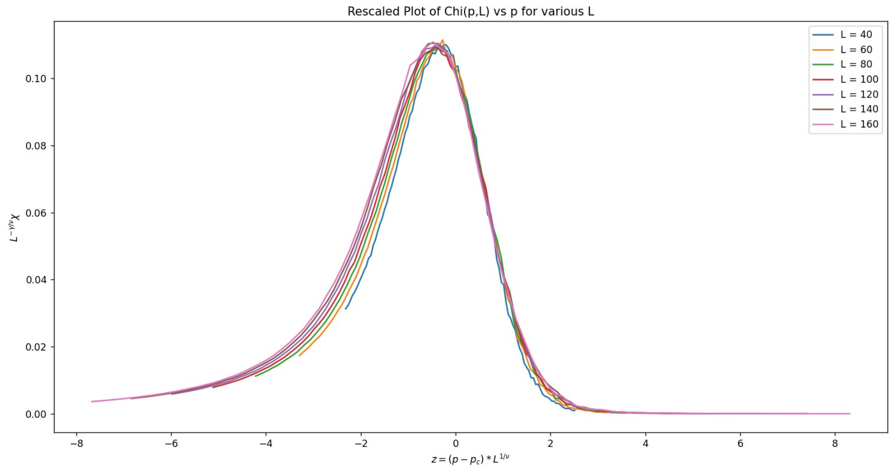

Figure 6: Average Cluster Size vs. Occupation Probability after rescaling using the exponents ν, γ/ν.

Binder Cumulant (U)

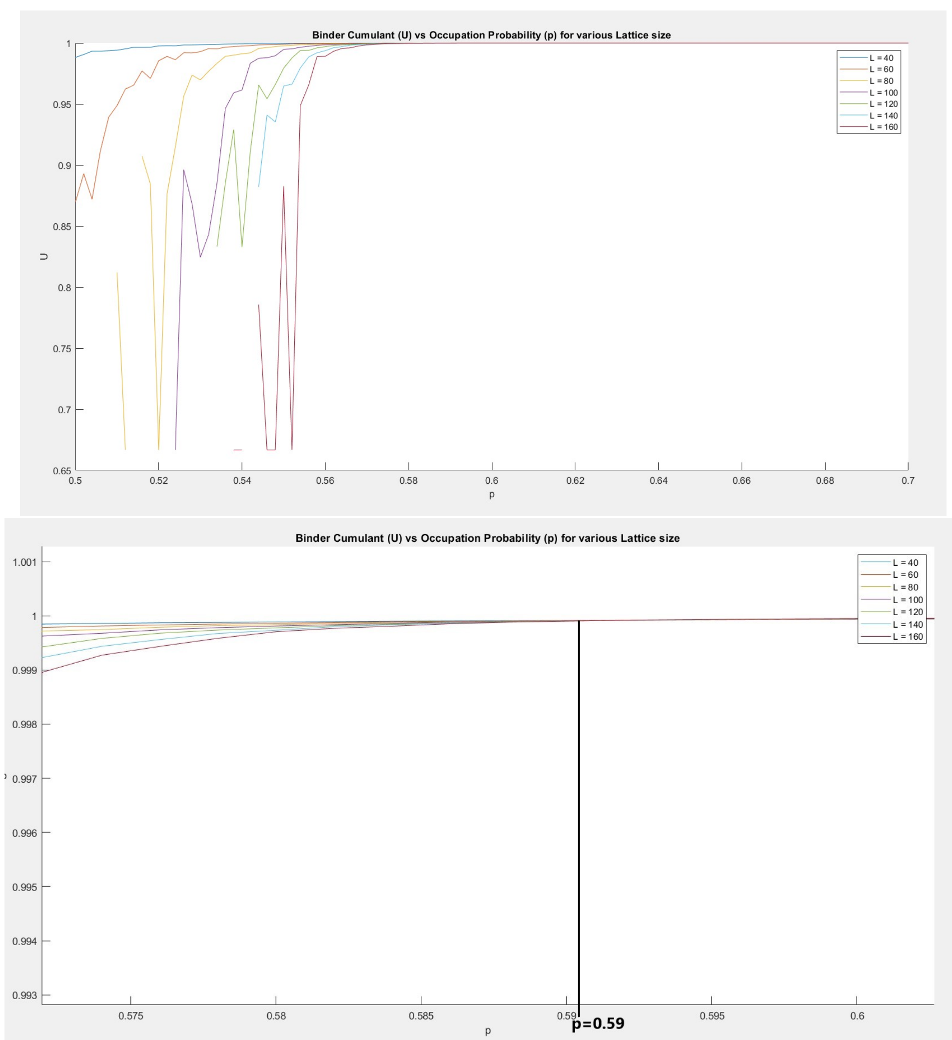

Figure 7: Binder Cumulant vs. Occupation Probability averaged over N=10000 iterations.

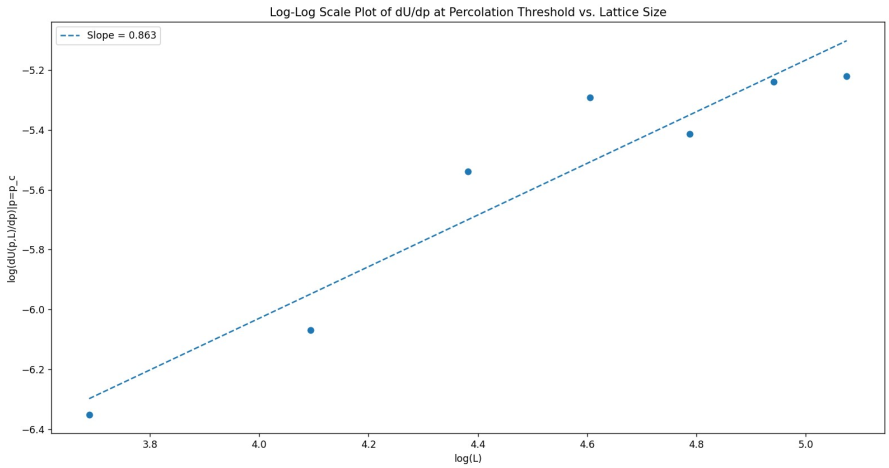

Figure 8: Slope of U wrt p at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent $1/\nu$

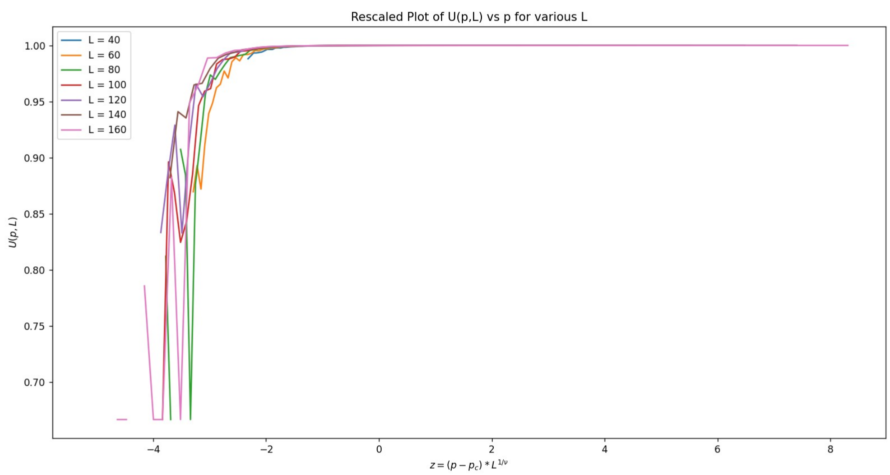

Figure 9: Binder Cumulant vs. Occupation Probability after rescaling using the exponent ν.

Discussion

- We know that the scaling form of P_∞ is given by

$$P_\infty(p, L) = L^{-\beta/\nu}\tilde{P}[(p - p_c)L^{1/\nu}]$$

  Thus, the plot of P_∞ vs p is as expected (figure 1). Looking at this form, we can conclude that at p = p_c

$$\tilde{P}[(p - p_c)L^{1/\nu}] = \tilde{P}[0] = cL^{-\beta/\nu}$$

  Hence, from the above equation and the graph, we can find the percolation threshold. Similarly by plotting log[P_∞] vs log[L], we can get the exponent β/ν.

- From figure 1 of P_∞, we can see that for all L values, the graphs converge at $p = 0.596$. Thus, the percolation threshold, $p_c = 0.596$.

- From figure 2, it can be seen that slope, $-\beta/\nu = -0.084$. Thus the value of $\beta/\nu = 0.084$.

- The scaling form of fluctuation in order parameter (average cluster size) is given by

$$\chi(p, L) = L^{\gamma/\nu}\tilde{\chi}[(p - p_c)L^{1/\nu}]$$

Hence, similar to the previous case, the percolation threshold and exponent γ/nu can be calculate from the average cluster size distribution vs p (Fisure 4).

- From figure 4 of χ, we can see that for all L values, the graphs have a peak at $p = 0.59$. This is very close to the percolation threshold, p_c found from P_∞ graph.

- From figure 5, it can be seen that slope, $\gamma/\nu = 1.758$.

- The scaling form of Binder Cumulant is given by,

$$U(p, L) = \tilde{U}[(p - p_c)L^{1/\nu}]$$

Hence, at p = p_c, the function is constant for all L. Therefore, we can calculate the percolation threshold from the intersection of U(p,L) for various L.

- By diffentiating Binder cumulant with respect to p, we get

$$\frac{dU(p, L)}{dp} = L^{1/\nu}\tilde{U}[(p - p_c)L^{1/\nu}]$$

Thus, at p = p_c, we can estimate the value of $1/\nu$ from the log[U] vs log[L] graph.

- From figure 7 of U, we can see that for all L values, the graphs have converge to 1 at $p = 0.59$. This is very close to the percolation threshold, p_c found from P_∞ graph.

- From figure 8, it can be seen that slope, $1/\nu = 0.863$. Hence, the exponent $\nu = 1.158$.

- From the above estimated exponents, we can see that,

$$\frac{\beta}{\nu} + \frac{\gamma}{\nu} = 0.084 + 1.758 = 1.842$$

We know that the scaling relation between these exponents is given by

$$\frac{\beta}{\nu} + \frac{\gamma}{\nu} = d$$

where d is the space dimension. In our case, the estimated value is 1.842, which is very close to the actual d =2.