

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, Evaluate the following:

- 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 .
- Plot $P_\infty(p, L)$ against p for various L and determine the value of β/ν . Verify the scaling form.
- Plot $\chi(p, L)$ against p for various L and determine the value of γ/ν . Verify the scaling form.
- Verify the Scaling relation between β/ν , γ/ν and the space dimension, d .

General Algorithm

- Step 1: Generate a $L \times L$ lattice. (Initially, $L=40$ and $p=0.5$)
- Step 2: Generate a random number, r , $L \times L$ 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 if 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 $P_\infty(p, L)$, $\chi(p, L)$, $P_\infty(p, L)^4$, $P_\infty(p, L)^2$.

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

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

- 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 $L=160$. Then 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 .

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

Hoshpep 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)

```
1 L = 40:20:160;
2 P = 0.5:0.002:0.7;
3 N=10000;
4 pinf = zeros(length(L),length(P));
5 csd = zeros(length(L),length(P));
6 bc = zeros(length(L),length(P));
7 a = zeros(length(L),length(P));
8 b = zeros(length(L),length(P));
9
10 for n = 1:N
11     for l = 1:length(L)
12         for p = 1:length(P)
13             lat = zeros(L(l),L(l));
14             %count=0;
15             for i = 1:L(l)
16                 for j = 1:L(l)
17                     r = rand;
18                     if r<P(p)
19                         lat(i,j) = 1;
20                         %count=count+1;
21                     end
22                 end
23             end
24             %disp(count/(L(l)*L(l)))
25             %disp(lat)
26             hk = zeros(L(l),L(l));
27             mass = [0];
28             k=1;
29             for i = 1:L(l)
30                 for j = 1:L(l)
31                     if i==1
32                         if j==1
33                             if lat(i,j)
34                                 hk(i,j) = k;
35                                 k=k+1;
36                                 mass = [mass,0];
37                                 mass(hk(i,j)) = mass(hk(i,j))+1;
38                                 continue;
39                             else
40                                 continue;
41                             end
42                         else
43                             if lat(i,j)
44                                 if lat(i,j-1)
45                                     hk(i,j) = hk(i,j-1);
```

```

46         mass(hk(i,j)) = mass(hk(i,j))+1;
47         continue;
48     else
49         hk(i,j) = k;
50         k=k+1;
51         mass = [mass,0];
52         mass(hk(i,j)) = mass(hk(i,j))+1;
53         continue;
54     end
55 end
56 end
57 end
58 if j==1
59     if lat(i,j)
60         if lat(i-1,j)
61             tmp = mass(hk(i-1,j));
62             tmp1= hk(i-1,j);
63             while tmp<0
64                 tmp1 = tmp;
65                 tmp = mass(abs(tmp1));
66             end
67             hk(i,j) = abs(tmp1);
68             mass(hk(i,j)) = mass(hk(i,j))+1;
69             continue;
70         else
71             hk(i,j) = k;
72             k=k+1;
73             mass = [mass,0];
74             mass(hk(i,j)) = mass(hk(i,j))+1;
75             continue;
76         end
77     end
78 end
79 if lat(i,j)
80     if lat(i-1,j)
81         if lat(i,j-1)
82             if hk(i,j-1) == hk(i-1,j)
83                 hk(i,j) = hk(i-1,j);
84                 mass(hk(i,j)) = mass(hk(i,j))+1;
85                 continue;
86             end
87             tmp = mass(hk(i-1,j));
88             tmp1= hk(i-1,j);
89             while tmp<0
90                 tmp1 = tmp;
91                 tmp = mass(abs(tmp1));
92             end
93             tmp = mass(hk(i,j-1));
94             tmp2= hk(i,j-1);
95             while tmp<0
96                 tmp2=tmp;
97                 tmp = mass(abs(tmp2));
98             end
99             if abs(tmp1)==abs(tmp2)
100                 hk(i,j) = abs(tmp1);
101                 mass(hk(i,j)) = mass(hk(i,j))+1;
102             else
103                 hk(i,j) = min(abs(tmp1),abs(tmp2));
104                 mass(hk(i,j)) = mass(abs(tmp1))+mass(abs(tmp2))+1;
105                 mass(max(abs(tmp1),abs(tmp2))) = -1*hk(i,j);
106             end
107         else
108             tmp = mass(hk(i-1,j));
109             tmp1=hk(i-1,j);
110             while tmp<0
111                 tmp1 = tmp;
112                 tmp = mass(abs(tmp1));
113             end
114             hk(i,j) = abs(tmp1);
115             mass(hk(i,j)) = mass(hk(i,j))+1;
116         end

```

```

117         elseif lat(i,j-1)
118             tmp = mass(hk(i,j-1));
119             tmp1= hk(i,j-1);
120             while tmp<0
121                 tmp1 = tmp;
122                 tmp = mass(abs(tmp1));
123             end
124             hk(i,j) = abs(tmp1);
125             mass(hk(i,j)) = mass(hk(i,j))+1;
126         else
127             hk(i,j) = k;
128             k=k+1;
129             mass = [mass,0];
130             mass(hk(i,j)) = mass(hk(i,j))+1;
131         end
132     end
133 end
134 end
135 %disp(hk)
136 s = [];
137 ns = [];
138 for t = 1:length(mass)
139     if mass(t)>0
140         if ismember(mass(t),s)
141             continue;
142         else
143             s = [s,mass(t)];
144             bla = 0;
145             for i = 1:length(mass)
146                 if mass(i)==mass(t)
147                     bla = bla+1;
148                 end
149             end
150             ns = [ns,bla];
151         end
152     end
153 end
154 %disp(s)
155 %disp(ns)
156 perc = [];
157 for i = 1:L(1)
158     if hk(1,i) ~=0
159         for j = 1:L(1)
160             if hk(L(1),j) == hk(1,i)
161                 if ismember(hk(1,i),perc)
162                     continue;
163                 else
164                     perc = [perc,hk(1,i)];
165                 end
166             end
167         end
168     end
169 end
170 %disp(perc)
171 massinf = zeros(length(perc),1);
172 tmp = 0;
173 for i = 1:length(perc)
174     massinf(i) = mass(perc(i));
175     tmp = tmp+massinf(i)/(L(1)*L(1));
176 end
177 for i = 1:length(s)
178     if ismember(s(i),massinf)
179         continue;
180     else
181         %tmp = tmp-s(i)*ns(i)/(L(1)*L(1));
182         csd(1,p) = csd(1,p)+s(i)*s(i)*ns(i)/(L(1)*L(1));
183     end
184 end
185 pinf(1,p) = tmp+pinf(1,p);
186 a(1,p) = a(1,p)+(tmp)^4;
187 b(1,p) = b(1,p)+(tmp)^2;

```

```

188         end
189     end
190     disp(n)
191 end
192 pinf = pinf/N;
193 csd = csd/N;
194 a = a;
195 b = b;
196 for l = 1:length(L)
197     for p = 1:length(P)
198         bc(l,p) = (1-(a(l,p)/(3*b(l,p)^2)));
199     end
200 end
201 %disp(pinf1)
202 figure(1);
203 title('Order Parameter (P_{inf}) vs Occupation Probability (p) for various Lattice
204     size')
205 xlabel('p')
206 ylabel('P_{inf}')
207 hold on;
208 for i = 1:length(L)
209     plot(P, pinf(i,:))
210 end
211 legend({'L = 40', 'L = 60', 'L = 80', 'L = 100', 'L = 120', 'L = 140', 'L = 160'})
212 hold off;
213
214 figure(2);
215 title('Average Cluster Size (Chi) vs Occupation Probability (p) for various Lattice
216     size')
217 xlabel('p')
218 ylabel('Chi')
219 hold on;
220 for i = 1:length(L)
221     plot(P, csd(i,:))
222 end
223 legend({'L = 40', 'L = 60', 'L = 80', 'L = 100', 'L = 120', 'L = 140', 'L = 160'})
224 hold off;
225
226 figure(3);
227 hold on;
228 title('Binder Cumulant (U) vs Occupation Probability (p) for various Lattice size')
229 xlabel('p')
230 ylabel('U')
231 for i = 1:length(L)
232     plot(P, bc(i,:))
233 end
234 legend({'L = 40', 'L = 60', 'L = 80', 'L = 100', 'L = 120', 'L = 140', 'L = 160'})
235 hold off;
236
237 writematrix(pinf, 'Pinf.csv')
238 writematrix(csd, 'csd.csv')
239 writematrix(bc, 'bc.csv')

```

Plots and Finding the Exponents(Python)

```

1 from cProfile import label
2 from operator import le
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import math as m
6
7 L = [40,60,80,100,120,140,160]
8 P = np.arange(0.5,0.701,0.002)
9 x = np.zeros((len(L)))
10 y = np.zeros((len(L)))
11
12 Pinf = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment
13     3\Pinf.csv",delimiter=",", dtype=float)
14 csd = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment
15     3\csd.csv",delimiter=",", dtype=float)
16 bc = np.loadtxt("D:\Academics\Engineering Physics\Sem-7\PH415\Assignments\Assignment

```

```

3\Bc.csv",delimiter=",", dtype=float)
15
16 pc = 0.596
17 Pinf_pc = Pinf[:,int((pc-0.5)/0.002)]
18 for i in range(0,len(L)):
19     y[i] = m.log(Pinf_pc[i])
20     x[i] = m.log(L[i])
21 plt.scatter(x,y)
22 z = np.polyfit(x,y, 1)
23 p = np.poly1d(z)
24 bn = -z[0]
25 plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
26 plt.title('Log-Log Scale Plot of P_inf at Percolation Threshold vs. Lattice Size')
27 plt.ylabel('log(P_inf(p=p_c,L))')
28 plt.xlabel('log(L)')
29 plt.legend()
30 plt.show()
31
32 chi_pc = csd[:,int((pc-0.5)/0.002)]
33 for i in range(0,len(L)):
34     y[i] = m.log(chi_pc[i])
35     x[i] = m.log(L[i])
36 plt.scatter(x,y)
37 z = np.polyfit(x,y, 1)
38 p = np.poly1d(z)
39 gn = z[0]
40 plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
41 plt.title('Log-Log Scale Plot of Chi at Percolation Threshold vs. Lattice Size')
42 plt.ylabel('log(Chi(p=p_c,L))')
43 plt.xlabel('log(L)')
44 plt.legend()
45 plt.show()
46
47 dU_pc = (bc[:,int((pc-0.5)/0.002)]-bc[:,int((pc-0.5)/0.002)-1])/0.002
48 for i in range(0,len(L)):
49     y[i] = m.log(dU_pc[i])
50     x[i] = m.log(L[i])
51 plt.scatter(x,y)
52 z = np.polyfit(x,y, 1)
53 p = np.poly1d(z)
54 nu = 1/z[0]
55 plt.plot(x, p(x), linestyle="--", label='Slope = '+str(np.round(z[0],3)))
56 plt.title('Log-Log Scale Plot of dU/dp at Percolation Threshold vs. Lattice Size')
57 plt.ylabel('log(dU(p,L)/dp)|p=p_c')
58 plt.xlabel('log(L)')
59 plt.legend()
60 plt.show()
61
62 for i in range(0,len(L)):
63     x = np.zeros((len(P)))
64     y = Pinf[i,:]
65     for j in range(0,len(P)):
66         x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
67         y[j] = y[j]*m.pow(L[i],bn)
68
69     plt.plot(x,y, label='L = '+str(L[i]))
70 plt.title('Rescaled Plot of P_inf(p,L) vs p for various L')
71 plt.xlabel(r"$z = (p-p_c)*L^{\{1/\nu\}}$")
72 plt.ylabel(r"$L^{\{\beta/\nu\}}P_{\infty}$")
73 plt.legend()
74 plt.show()
75
76 for i in range(0,len(L)):
77     x = np.zeros((len(P)))
78     y = csd[i,:]
79     for j in range(0,len(P)):
80         x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
81         y[j] = y[j]*m.pow(L[i],-gn)
82
83     plt.plot(x,y, label='L = '+str(L[i]))
84 plt.title('Rescaled Plot of Chi(p,L) vs p for various L')

```

```

85 plt.xlabel(r"$z = (p-p_c)*L^{\{1/\nu\}}$")
86 plt.ylabel(r"$L^{-\{\gamma/\nu\}}\chi$")
87 plt.legend()
88 plt.show()
89
90 for i in range(0,len(L)):
91     x = np.zeros((len(P)))
92     y = bc[i,:]
93     for j in range(0,len(P)):
94         x[j] = (P[j]-pc)*m.pow(L[i],1/nu)
95         y[j] = y[j]
96
97     plt.plot(x,y, label='L = '+str(L[i]))
98 plt.title('Rescaled Plot of U(p,L) vs p for various L')
99 plt.xlabel(r"$z = (p-p_c)*L^{\{1/\nu\}}$")
100 plt.ylabel(r"$U(p,L)$")
101 plt.legend()
102 plt.show()

```

Results

Order Parameter (P_∞)

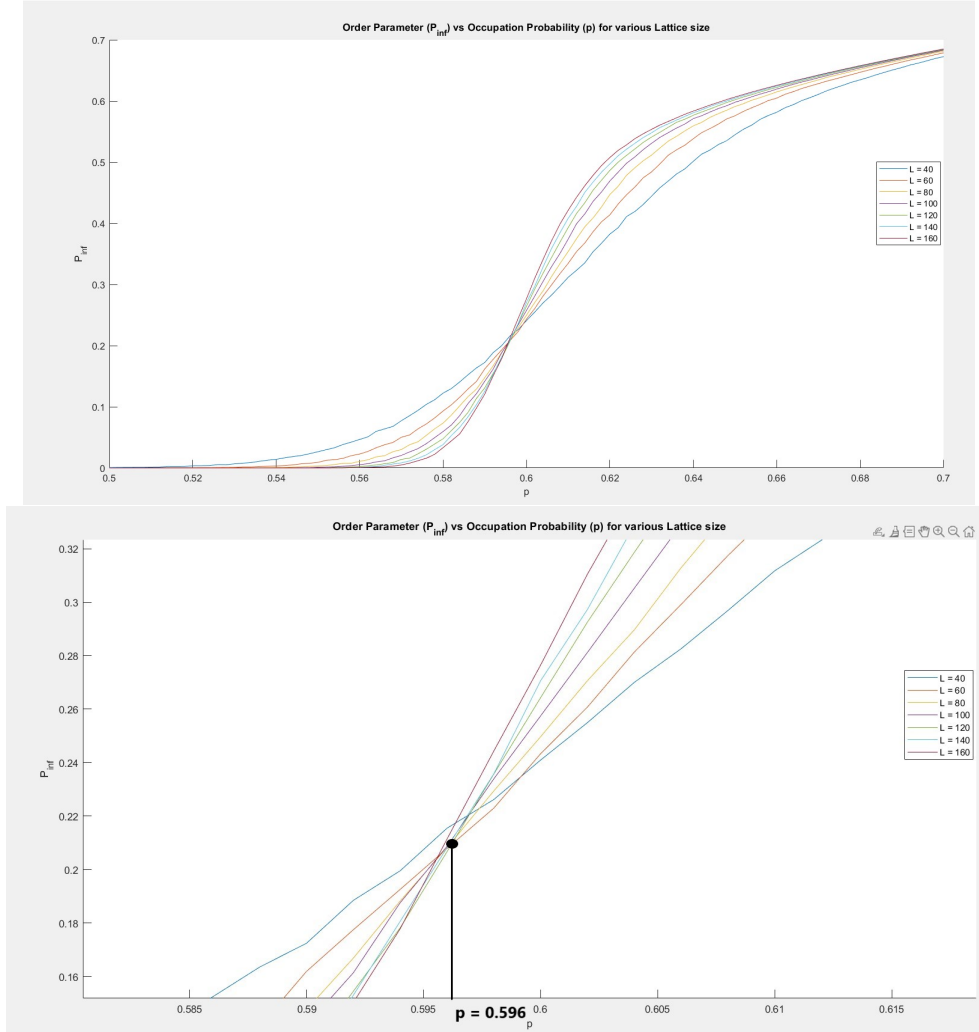


Figure 1: Order Parameter vs. Occupation Probability averaged over N=10000 iterations.

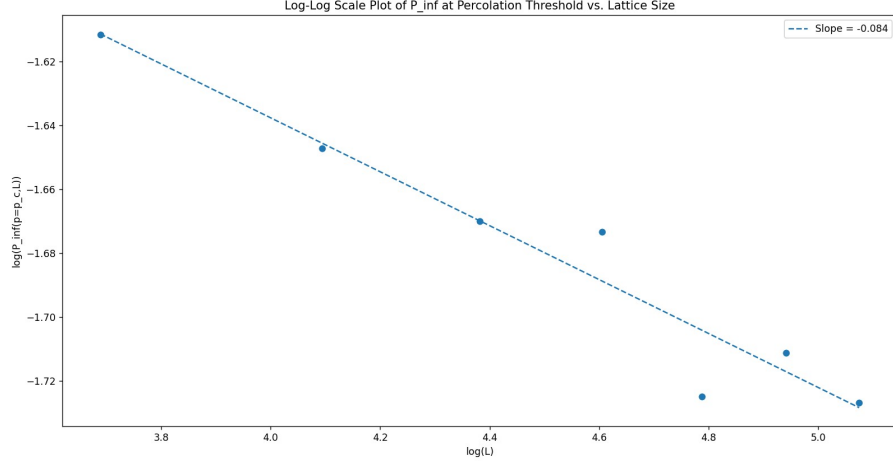


Figure 2: Order Parameter at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent β/ν

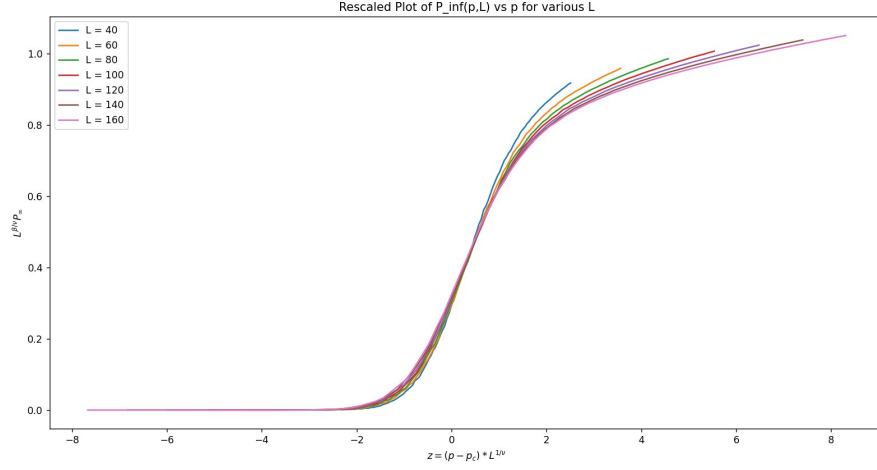


Figure 3: Order Parameter vs. Occupation Probability after rescaling using the exponents ν , β/ν .

Average Cluster Size (χ)

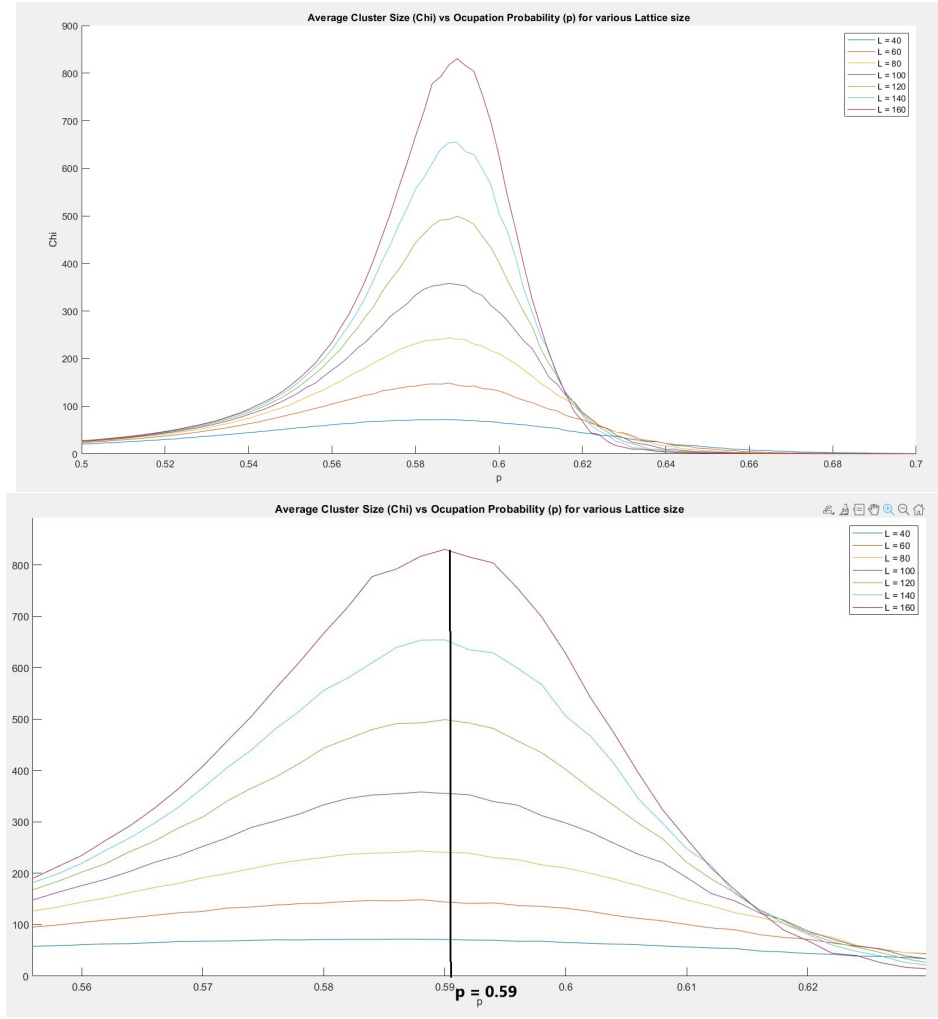


Figure 4: Average Cluster Size vs. Occupation Probability averaged over $N=10000$ iterations.

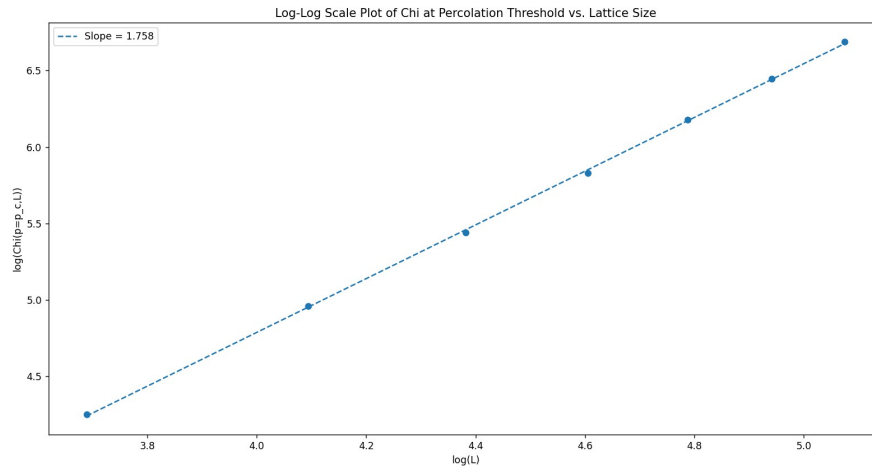


Figure 5: Average Cluster Size at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent γ/ν

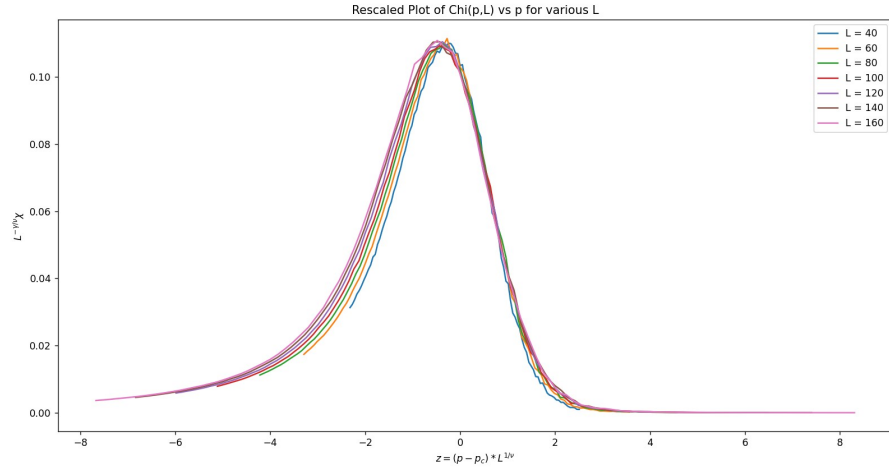


Figure 6: Average Cluster Size vs. Occupation Probability after rescaling using the exponents ν , γ/ν .

Binder Cumulant (U)

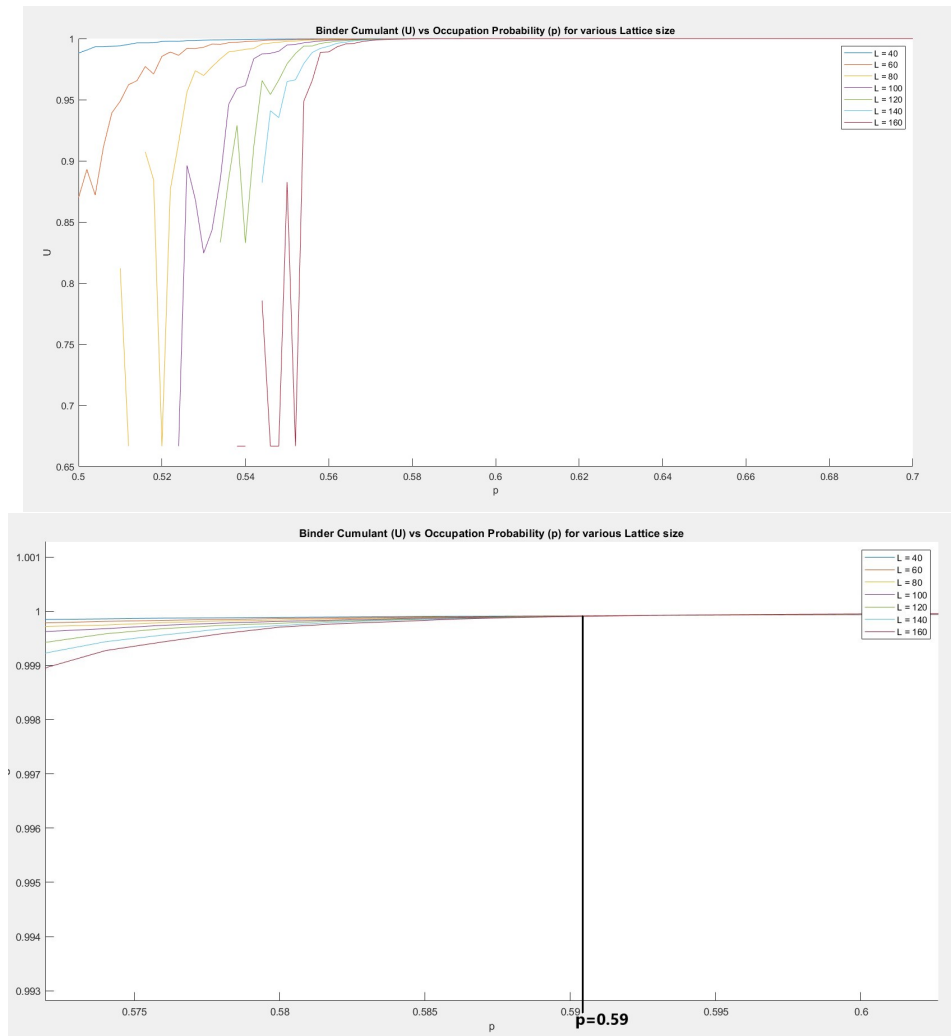


Figure 7: Binder Cumulant vs. Occupation Probability averaged over $N=10000$ iterations.

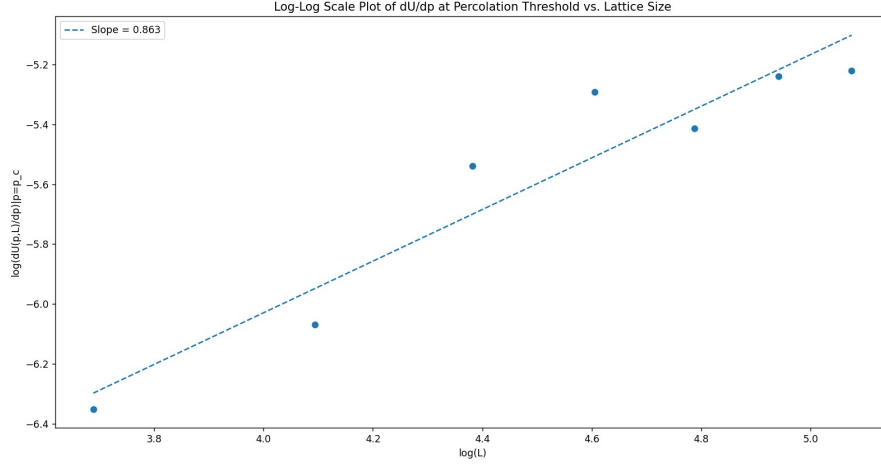


Figure 8: Slope of U wrt p at Percolation Threshold vs. Lattice Size in log-log scale to find the slope and hence the exponent $1/\nu$

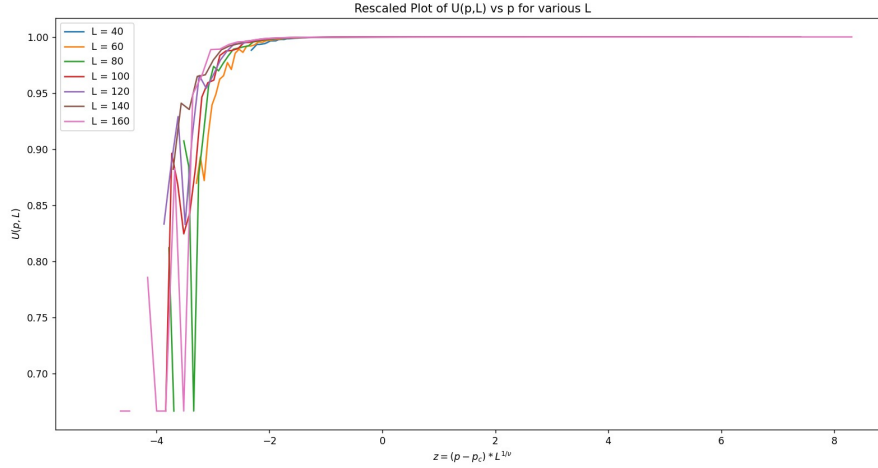


Figure 9: Binder Cumulant vs. Occupation Probability after rescaling using the exponent ν .

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 $\log[P_\infty]$ 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/ν 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 $d = 2$.