20 Probability

20.1 The Percolation Model

**Question 1:**

1. Why θ is non-decreasing in ?

According to the notes at the end of the project, it suggests that we may label each edge with a pseudo-random number which is uniformly distributed from 0 to 1 and use the same realization for all values of simultaneously. We assume an edge is open when the assigned number of the edge is .

Let be the set of edges that are open under the and ,  
Under the same realization, we can deduce that if   
As there are more edges available for than for (for which ), we can conclude that  
.  
Therefore, if .

1. Show is decreasing in .

For , we can see it is dependent on as we know if , must be true as well.

Therefore,

■

1. Give an estimate for the likely size of the error .

We use Central Limit Theorem in this question.  
Central Limit Theorem stated that:

I suggest and

Hence,

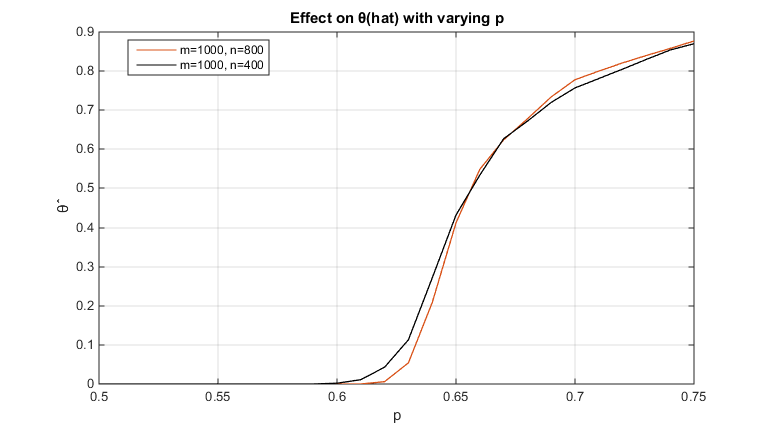
For a fixed , we can reduce the error of by increase m. The order of the reduction in error will be in order

■

**Question 2:**

Regarding to the computational time of the program, I have chosen (m=1000, n=800) and (m=1000, n=400) for my program [percolationQ2\_mod.m] to plot against for , which takes around 41 seconds to run for , which is close to the our believed result, .

Graph 1: Relationship between

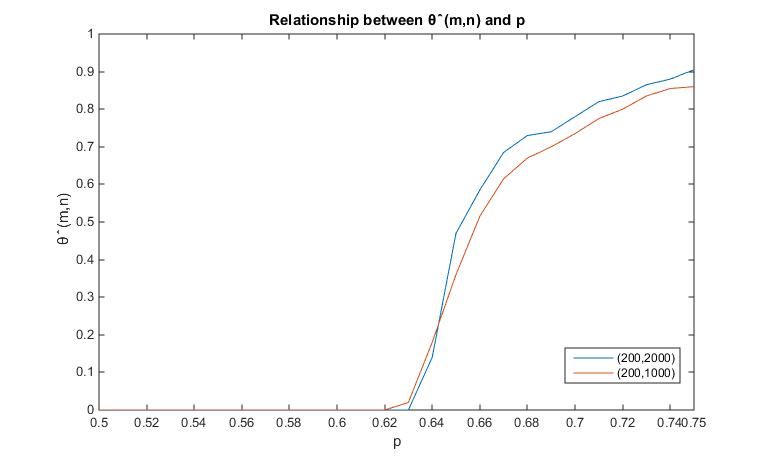


In my program [percolationQ2\_mod.m], I have first generate 2 random matrix of size representing the values on the vertical and horizontal edges from node. It runs at . Secondly, I followed the suggested algorithm from the notes and calculate for all nodes on the grid which involves 3 comparison operations per node which is running at as well. Thirdly, I check for any for and for all in the given range and this action works in . And lastly, we repeat the above for different realizations which means the complexity of the program in total will be We can deduce that the computational time for the program is also in

Therefore, I have chosen a smaller value for compared with at the start of the question.

As it is more likely for when for p fixed, we expect increases and getting closer to 0.644 as n increases and the graph of shifts to the right compared with the graph we have above.

Graph 2: plotting



Comparing two graphs with in Graph 2, we can see shifts from 0.62 to 0.63, which is closer to the true value, 0.644 as n increases. In addition, we can see the graph behaves like the error function and tends to a Heaviside function as n increases. I believe as n increases, the graph will tend to where H is a Heaviside function.

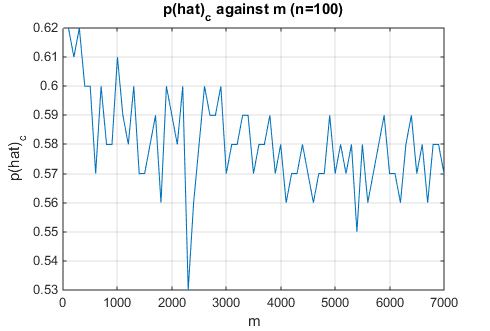
**Question 3:**

From graph 4, we can see increases and tends to reach the true value as increases and (large) fixed. This is because, similar reason in Question 2, it is more likely for . Therefore, the value for increases as n increases. On the other hand, as is a decreasing function in n and increasing function in p, for getting the for fixed m and increasing n, we should expect increases and it will always be and for .

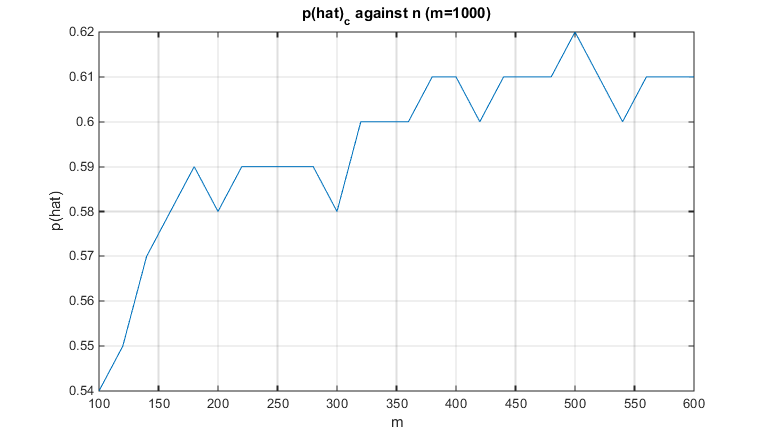
For fixed n, as m increases, we expect fluctuate and converges to a limit where is a constant and (‘Less than or equal to because of squared statement above.). For showing the convergence of , we may use the statement from the law of large numbers where the sample average tends to the expected value as . We can confirm our prediction with the Graph 3 as we can see a fluctuation between 0.5 to 0.57 and converging to the range of [0.51, 0.54] where is.

From both graph, as we achieve our approximation by averaging m realizations with 2 pseudo-random matrices, it is normal to see fluctuation, however, we can still see the tendency of the graph.

Graph 3: plotting



Graph 4: plotting



**Question 4:**

To estimate , I have written a program [Q4.m] which gives me the out of the m realization with a given p through running the program [percolationQ2\_mod.m] and calculate with the calculated and input n.

It is tricky to pick the suitable value for m and n as it always turns out to be undefined if I have picked a relatively large n due to if n is large and p is in the range [0.3-0.6] and does not give us any useful information at all. Therefore, I need to pick a relative small n for this estimation.

On the other hand, as there is more chance for me to get at least a realization with if I picked a large m, I expect to pick a large m in respect to n for the estimation.

As an increasing function in p and decreasing function in n, I expect that we can increase our value of n as we increase in p. By trial and error, I obtained the largest n which works for the smallest p for a large m=100000. To avoid unreasonably long computation time, as the program in , I have made adjustment of the value of m according to the value of n.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | m | n |
| 0.3 | 0.5207 | 100000 | 20 |
| 0.4 | 0.3005 | 50000 | 36 |
| 0.5 | 0.1012 | 30000 | 95 |
| 0.6 | 0.0186 | 10000 | 250 |

In theory, I expect to see decreases as p is reaching from below. This is because if p is higher so we expect as p increases.

**Reference:**

1. <http://www.stat.ucla.edu/~nchristo/introeconometrics/introecon_central_limit_theorem.pdf>  
   ‘The Central Limit Theorem’

**Program:**

1:

function [W]=percolationQ2\_mod(m,n,pstart,pend)

%m is the number of repeats; n is value of how far we

%approximate)

%p start and p end indicates range of p for the graph and approximation

tic;

W=[];

for y=1:m,

%generate different matrix first

map=rand(n,n,2);

T=nan(n,n);

T(1,1)=0;

%1 is horizontal 2 is vertical

%coord in T is i+1,j+1

%define boundary condition

for t=2:n,

T(1,t)=max(T(1,t-1),map(1,t-1,1));

T(t,1)=max(T(t-1,1),map(t-1,1,2));

end

%calculate the probability of each node being accessible

for k=2:n,

for u=2:(n-k+1)

ver=max(T(k-1,u),map(k-1,u,2));

hor=max(T(k,u-1),map(k,u-1,1));

T(k,u)=min(ver,hor);

end

end

O=0;

l=0;

e=[];

%check any nodes at distance n being accessible for different p

for p=pstart:0.01:pend,

l=l+1;

e=[e;p];

for z=1:n,

if T(z,n-z+1)<=p

O=1;

end

end

W(l,y)=O;

end

end

W=sum(W,2)./m;

toc;

plot(e,W)

end

2:

function Q3fixm\_mod(m)

T=[];

for n=100:20:600,

W=percolationQ2\_mod(m,n,0.5,0.7);

W=sum(W,2);

K=find(W);

P=0.5+(K(1)-1)\*0.01;

T=[T;n,P];

end

plot (T(:,1),T(:,2))

3:

function Q3fixn\_mod(n)

T=[];

for m=100:100:7000,

W=percolationQ2\_mod(m,n,0.45,0.75);

W=sum(W,2);

K=find(W);

P=0.5+(K(1)-1)\*0.01;

T=[T;m,P];

end

plot (T(:,1),T(:,2))

4:

function [K]=Q4(m,n,p)

[W]=percolationQ2\_mod(m,n,p,p);

K=(log(W(1)))/n;

end