**Name:** Duc Trinh

**Project Advisor:** Dániel Ábel

Term Project: Percolation transition of the E-R graph

1. **Introduction:**

The project aims to study the percolation transition of the E-R graph numerically. In this project, I will examine the scaling of the size of the largest connected component with the linking probability *(p)* at the critical point. At the same time, I also want to examine the scaling of the size of the largest connected component with the system size *(N)* at the critical point. The data was all collected through networkx ran on either Jupyter Notebook or Google Colab.

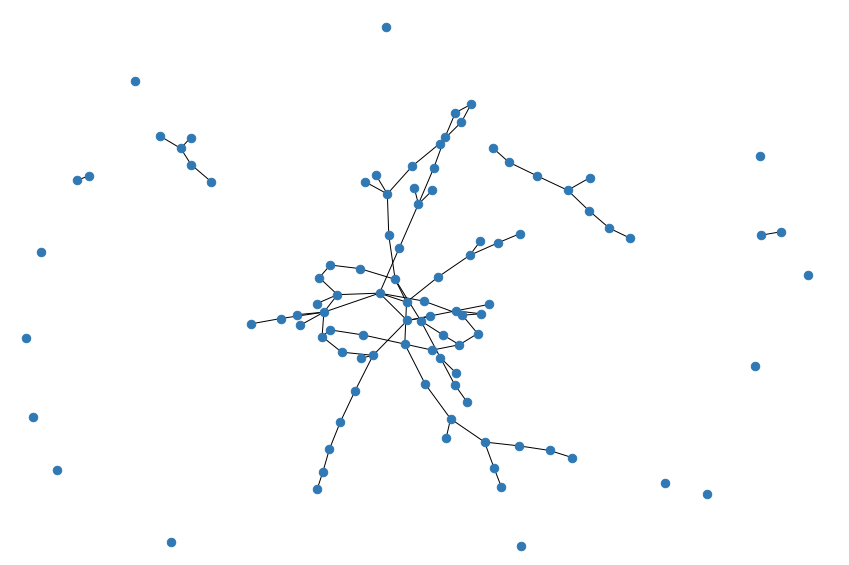
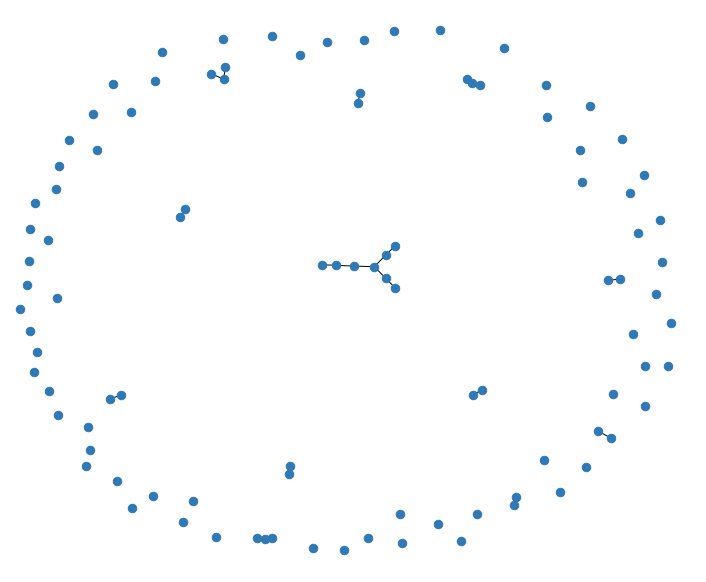
1. **Relevant concepts**
2. **The Erdos-Rényi model**

The Erdos-Rényi (E-R) model is the model of interest in my study. It was introduced by Pál Erdos and Alfréd Rényi in 1959. To generate an E-R model, one takes *N* nodes, and then uniformly link each pair of nodes independently with probability *p.* Since the E-R model is the only model I am using for my project, I want to denote that if I mention a network *G(N,p)*, then I am referring to a network created using the E-R model. One important property of the E-R model that is going to come up in this project is the average degree *(<k>)* of G(N,p) is equal to *(N-1)p*.

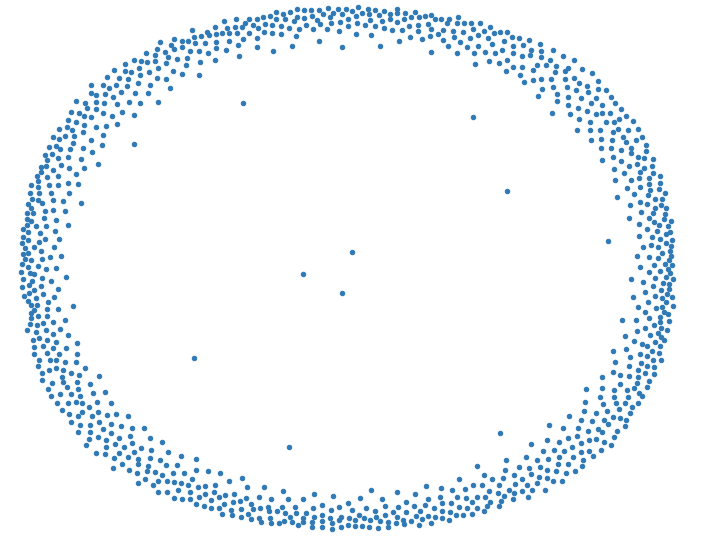
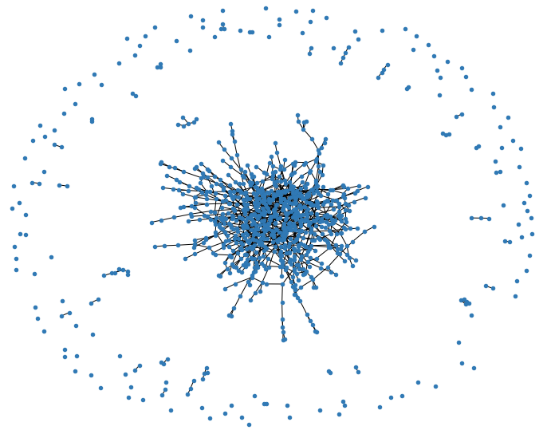
1. **Giant component**

A component is a group of nodes that are connected to each other in a network. A network is considered to be functioning when it has a giant component, which implies that most nodes are reachable from other nodes in the network.

Pictures of E-R graphs without and with a giant component:



G(100, 0.0051) G(100, 0.0151)



G(1000, 0.002)

G(1000, 0.000001)

The relative size of the largest component is denoted as with *s­1* as the size of the largest component in the network. Through theoretical work, it can be shown that . Note that this result means we expect *S* to behave like a power-law function.

The ***transition point*** in a network is where *S* becomes larger than zero. Whenever we have , then we have a giant component in an E-R graph.

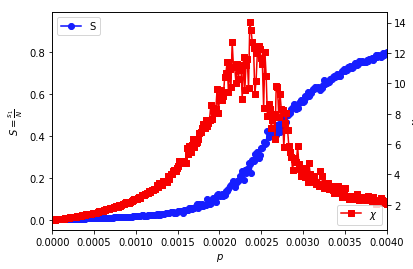
1. **Percolation transition**

The percolation transition of the E-R graph is the transition between two states of any E-R graph: without and with a giant component. If we take *p* as the control parameter, then this transition takes place at some critical value for *p*, which will be denoted as *pc*. Likewise, if we take *N* as the control parameter, then this transition takes place at some critical value for *N*, which will be denoted as *Nc*. According to theoretical works, we know that the percolation transition takes place at , which means and .

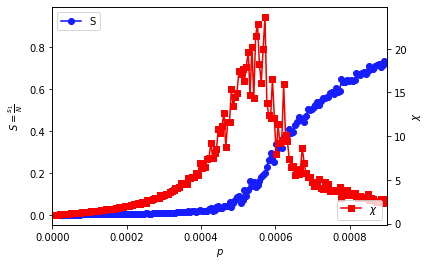
1. **Susceptibility**Susceptibility, usually denoted by , is the measurement of the sensitivity of the network in regard to outside change that can drive the system towards the other phase. Hence, susceptibility is strongest near the neighborhood of the transition point, and it peaks at the transition point. For this project, I am estimating the critical point based on where peaks. My main interest lies in the behavior of *S* in the surrounding neighborhood of this estimated critical point.
2. **Findings**
3. **Scaling of *S* in regard to *p***

Method to gather data: First, I pick a system size *N* to work with, and I calculate its *pc* using the theoretical result. Then, I estimate and take samples from the neighborhood around the critical point as With experimentation, it occurs to me that the bigger the system size, the more abrupt the transition takes place. Hence, for bigger system, I need to decrease , and increases the number of samples I take from the neighborhood to make sure I do not miss the peak. Finally, I generate multiple samples E-R graphs with *N* and the samples *p* from the neighborhood I decided on to calculate their average *S* and . An issue that comes up here is that small system tends to have high variance in their *S* and , so I need to generate more sample E-R graphs for smaller system.

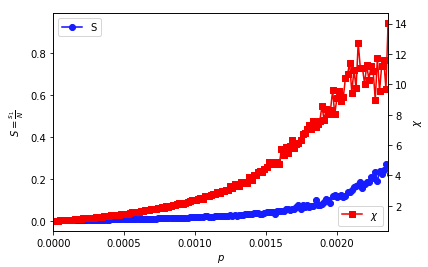
Note that the estimated *pc* is decided based on where the peak in takes place. Here are some graphs I produced from the gathered data:



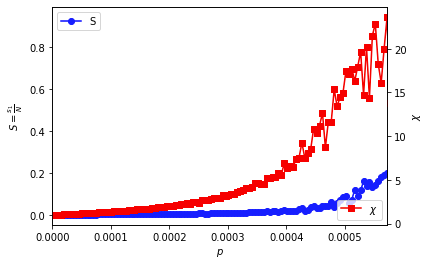
*S* and graph for G(N = 500, p)



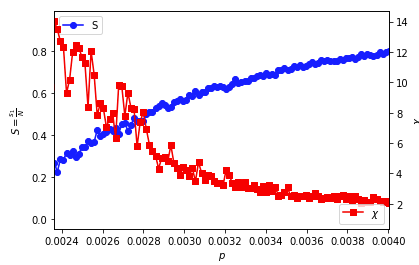
*S* and graph for G(N = 2000, p)



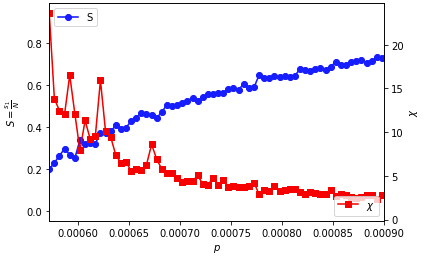
*S* and graph for G(N = 500, p) before the estimated *p­c*



*S* and graph for G(N = 2000, p) before the estimated *p­c*



*S* and graph for G(N = 500, p) after the estimated *p­c*



*S* and graph for G(N = 2000, p) after the estimated *p­c*

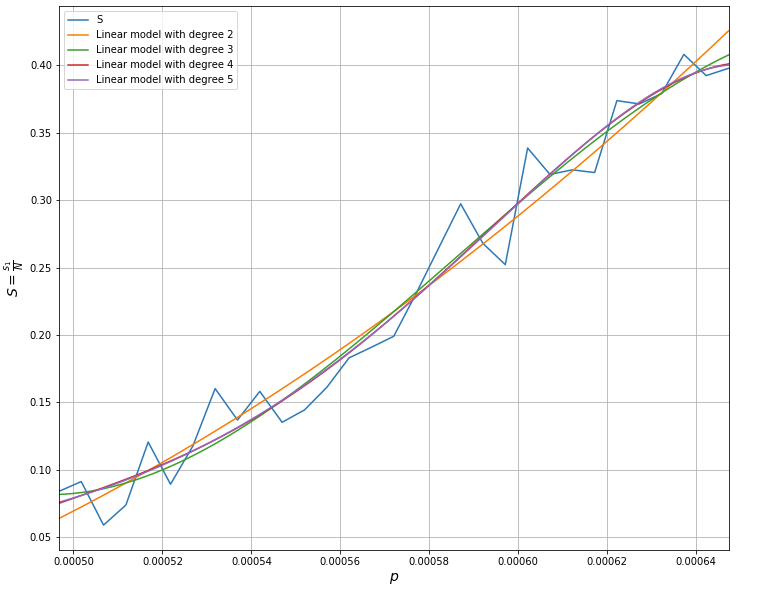
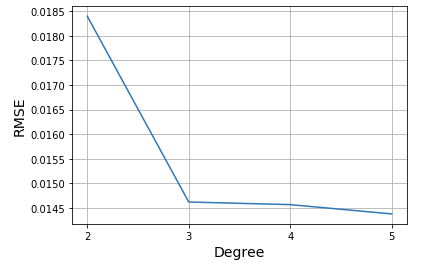
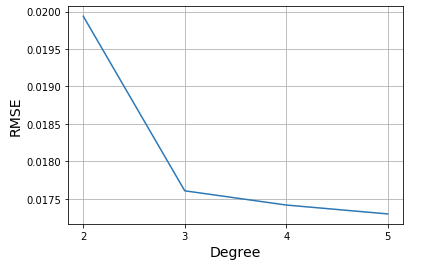
The estimated *pc* for G(N = 500, p) is roughly 0.0024, slightly bigger than the theoretical *pc* which is around 0.002. The estimated *pc* for G(N = 2000, p) is roughly 0.00057, also slightly bigger than the theoretical *pc* which is around 0.0005. We can see that our estimates of *pc* got really close to the theoretical values.

For both graph size, we see *S* begins to increase near the estimated critical point and continues to increase pass the estimated critical point. The value of *S* at the estimated critical point for G(N = 500, p) is 0.27 which is slightly bigger than that of G(N = 2000, p) which is 0.20.

On the other hand, we also see increases near the estimated critical point, even before *S* begins to increase. As expected, after the estimated critical point, we start to see decreases.

We see some further differences with the two different graph sizes here. Peak for G(N = 500, p) is 14.03 which is smaller than that of G(N = 2000, p) which is 23.60. We also see that the neighborhood around the critical point is bigger for G(N = 500, p) (from 0.0010 to 0.0035 ~ range of 0.0025) than that of G(N = 2000, p) (from 0.0003 to 0.0007 ~ range of 0.0004). Therefore, it seems like increases at a higher rate for bigger system, but it also decreases at a higher rate after the critical point.

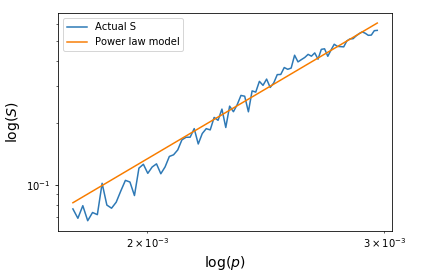
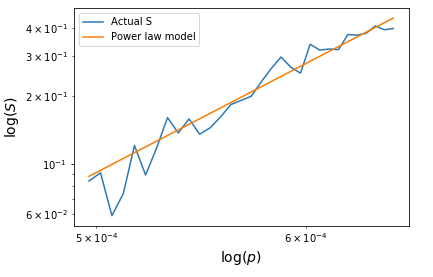
Next, I try to fit some simple functions to the neighborhood around the critical point to see if the scaling of *S* in respect to *p* behaves similar to any of these functions. I try to fit the data using linear models’ degree 2 to 5, and then I use a power-law fit. Here, I measure the performance of the models using their root mean square errors (RMSE).

Fitting *S* in regard to *p* for G(N = 2000, p) with linear models

Linear model degree versus error for G(N = 2000, p)

Linear model degree versus error for G(N = 500, p)

We can see that the linear models perform quite well here with quite a low error rate even from degree 2. The error rate naturally decreases as the degree of the model increases.



Power-law fit for G(N = 2000, p)

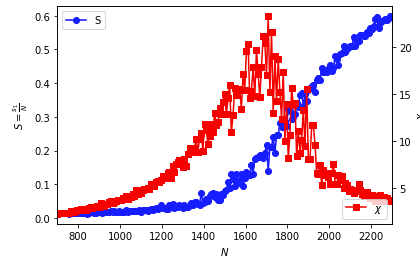
Power-law fit for G(N = 500, p)

The power-law fit also works well for both G(N = 500, p) and G(N = 2000, p). Note that from the theoretical works, we know that *S* should behaves like a power-law function. However, the power-law fits actually performed worse than the linear models. The RMSE for G(N = 500, p) is 0.024, and the RMSE for G(N = 2000, p) is 0.027. Hence, they both lost to their respective second-degree linear model by a small margin. The exponent of the power-law fit for G(N = 500, p) is 3.84 while that of G(N = 2000, p) is 5.66. I tried the power-law fit with G(N = 1000, p) and got an exponent of 4.78, so it seems the exponent scales in the same direction with *N*.

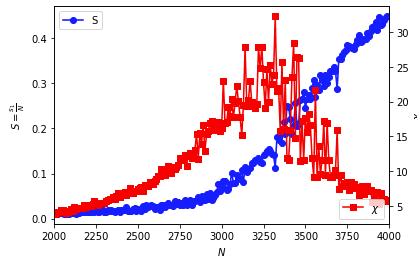
1. **Scaling of *S* in regard to *N***

Method to gather data: First, I picked a *Nc* to work with, and I calculate its *p* using the theoretical result. I choose *Nc* first since it is easier to work with an integer *N.* Then, I estimate and take samples from the neighborhood around the critical point as With experimentation, it seems to me that the bigger *Nc*, the bigger the neighborhood around *Nc­* needs to be. Hence, I increased for bigger *N­c­*. Finally, I generate multiple samples E-R graphs with *p* and the samples *N* from the neighborhood I decided on to calculate their average *S* and . The same issue persists with small system having high variance in their S and χ, so I need to generate more sample E-R graphs for smaller system.

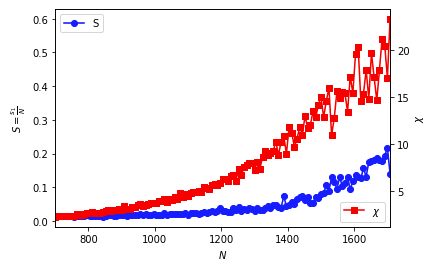
Note that the estimated *Nc* is decided based on where the peak in takes place. Here are some graphs I produced from the gathered data:



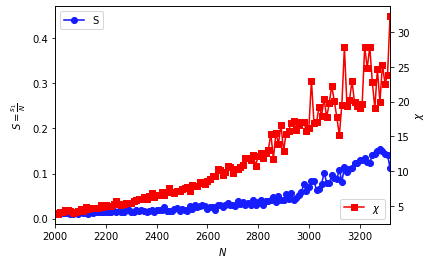
*S* and graph for G(N, p = 0.00067)



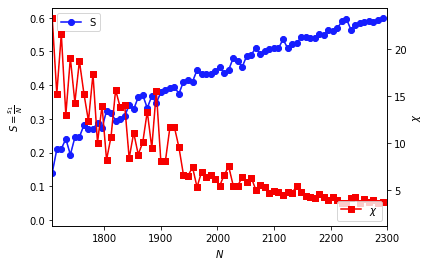
*S* and graph for G(N, p = 0.00033)



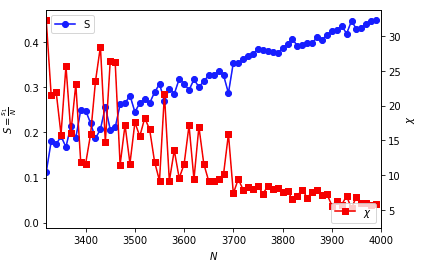
*S* and graph for G(N, p = 0.00067) before the estimated *N­c*



*S* and graph for G(N, p = 0.00033) before the estimated *N­c*



*S* and graph for G(N, p = 0.00067) after the estimated *N­c*



*S* and graph for G(N, p = 0.00033) after the estimated *N­c*

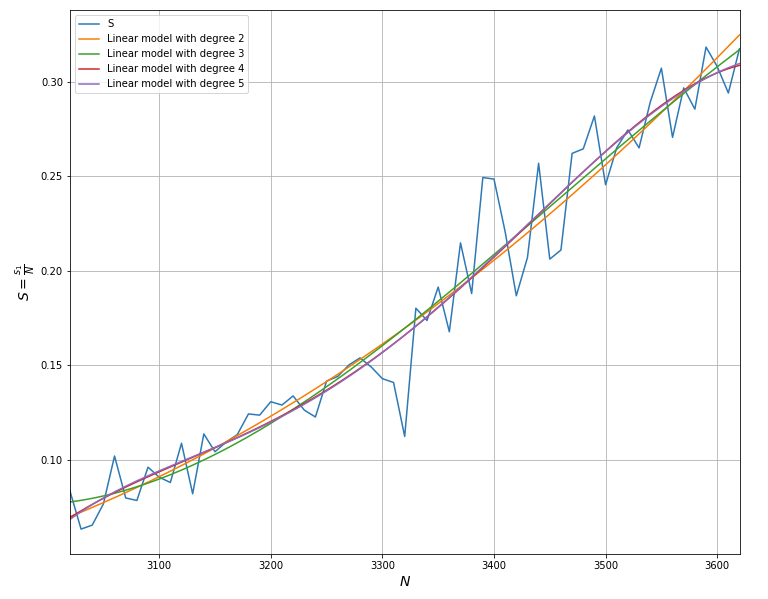
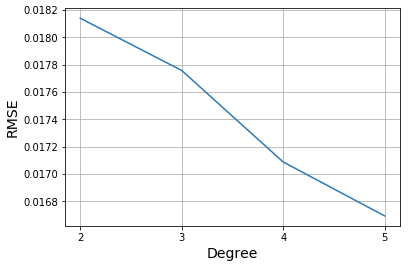
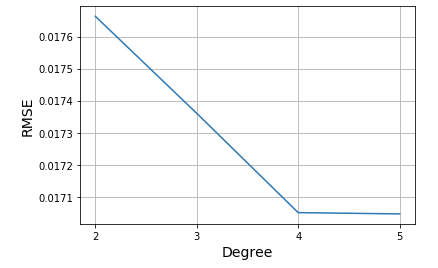
The estimated *Nc* for G(N, p = 0.00067) is 1708, bigger than the theoretical *Nc* which is 1500. The estimated *Nc* for G(N, p = 0.00033) is 3320, also bigger than its theoretical *Nc* which is 3000. We can see that our estimates of *Nc* got decently close to the theoretical values.

For both graph size, we see *S* begins to increase near the estimated critical point and continues to increase pass the estimated critical point. The value of *S* at the estimated critical point for G(N, p = 0.00067) is 0.14 which is slightly bigger than that of G(N, p = 0.00033) which is 0.11.

On the other hand, we also see increases near the estimated critical point, even before *S* begins to increase. As expected, after the estimated critical point, we start to see decreases.

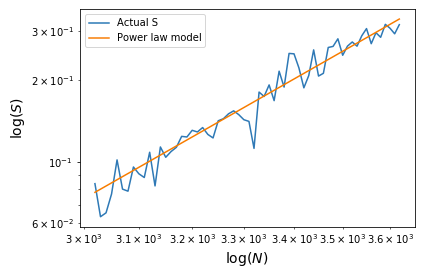
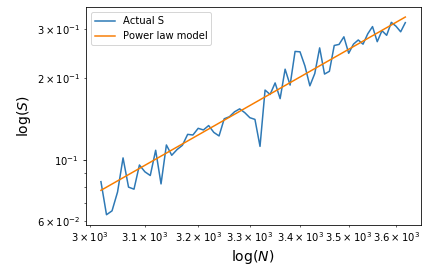
We see some further differences with the two different graph sizes here. Peak for G(N, p = 0.00067) is 23.25 which is smaller than that of G(N = 2000, p) which is 32.27. The neighborhood around the critical point for G(N, p = 0.00067) (from 900 to 2100 ~ range of 1200) is slightly bigger than that of G(N, p = 0.00033) (from 2750 to 3750 ~ range of 1000). Therefore, it seems like increases at a higher rate for bigger system, but it also decreases at a higher rate after the critical point.

Next, I try to fit some simple functions to the neighborhood around the critical point to see if the scaling of *S* in respect to *p* behaves similar to any of these functions. I try to fit the data using linear models’ degree 2 to 5, and then I use a power-law fit. Here, I measure the performance of the models using their root mean square errors (RMSE).

 Fitting *S* in regard to *p* for G(N, p = 0.00033) with linear models

Linear model degree versus error for G(N, p = 0.00033)

Linear model degree versus error for G(N, p = 0.00067)

We can see that the linear models perform quite well here with quite a low error rate even from degree 2. The error rate naturally decreases as the degree of the model increases.

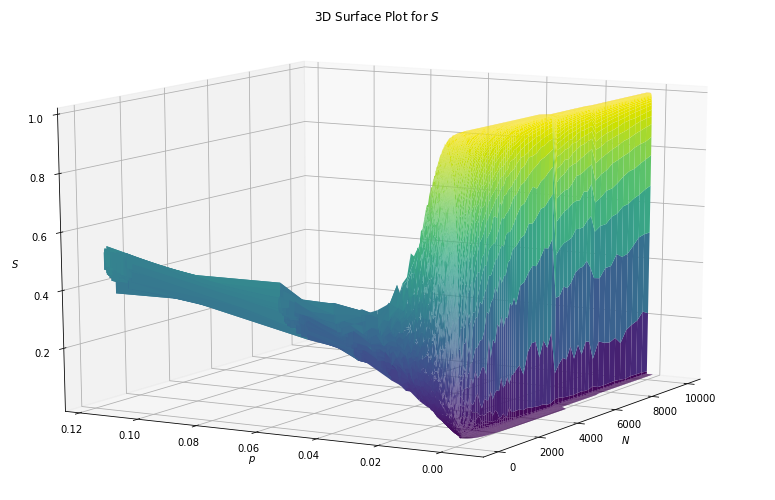
Power-law fit for G(N, p = 0.00033)

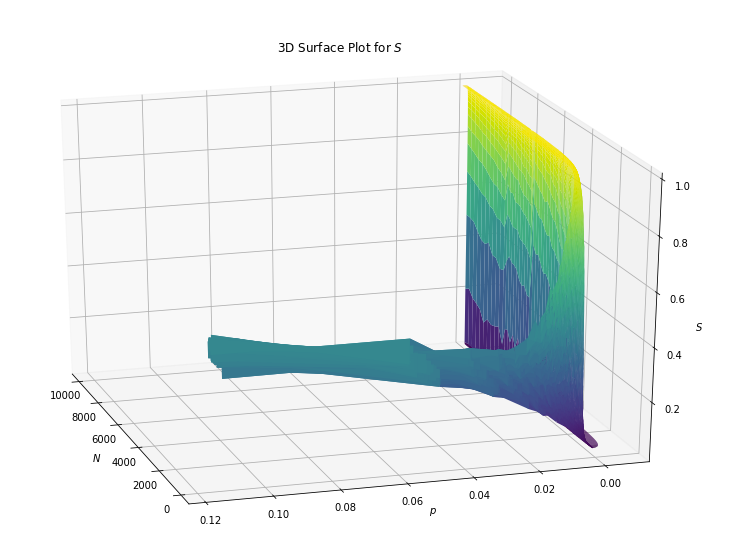
Power-law fit for G(N, p = 0.00067)

The power-law fit also works well for both G(N, p = 0.00067) and G(N, p = 0.00033). Note that from the theoretical works, we know that *S* should behaves like a power-law function. However, the power-law fits actually performed worse than the linear models. The RMSE for G(N, p = 0.00067) is 0.0205, and the RMSE for G(N, p = 0.00033) is 0.0183. Hence, they both lost to their respective second-degree linear model by a small margin. The exponent of the power-law fit for G(N, p = 0.00067) is 5.94 while that of G(N, p = 0.00033) is 8.038. I tried the power-law fit with G(N, p = 0.001) and got an exponent of 6.80, so it seems the exponent scales in the same direction with *p*.

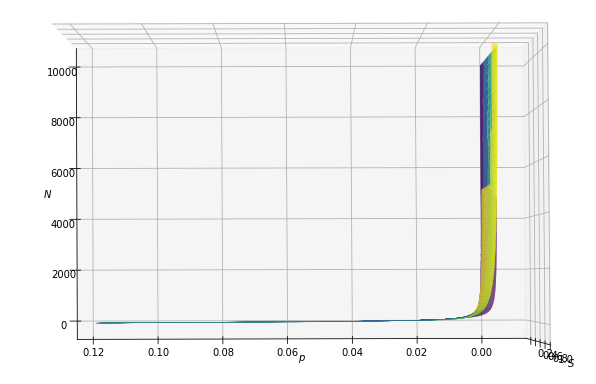
1. **Studying G(N,p) as a multivariable function**

Method: I basically repeat the steps I did to perform part 1, but I do it with a lot more system sizes (ranging from 10 to 10000). The most difficult part was to tune for the different behavior that comes for each system size. As I have mentioned before, the small systems have a lot more variability, and require ones to make more sample graphs. On the other hand, the big systems have narrower peaks, which requires one to take more samples in the neighborhood around the critical point. I broke the data gathering for this part into four different intervals to try to adjust for that. Luckily, matplotlib does allow for the different spacing of samples in system sizes. However, I needed to keep the number of samples in *p* the same, so what I did was to decrease as the system size increases. The last interval (from 7000-10000) ran the slowest and was the one that I had to reran and retuned the most (5 times in total). Most of the data gathering was done over Google Colab to reduce the stress on my computer. The only issue was Google Colab was unreliable in saving my data. I needed to be there to rerun the saving command when it finishes running. Hence, when I run my code overnight, I usually run it locally. Here are the plots I received from gathering the data:



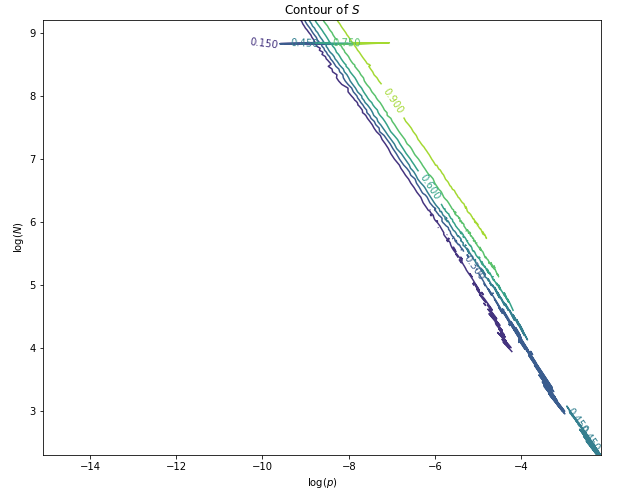


We can see that a cut of the 3D surface plot such as G(N = x, p) is going to give us the same *S* curve as the one we can see in part 1. Unfortunately, it is hard to see the cut for G(n, p = y) here due to insufficient sampling.

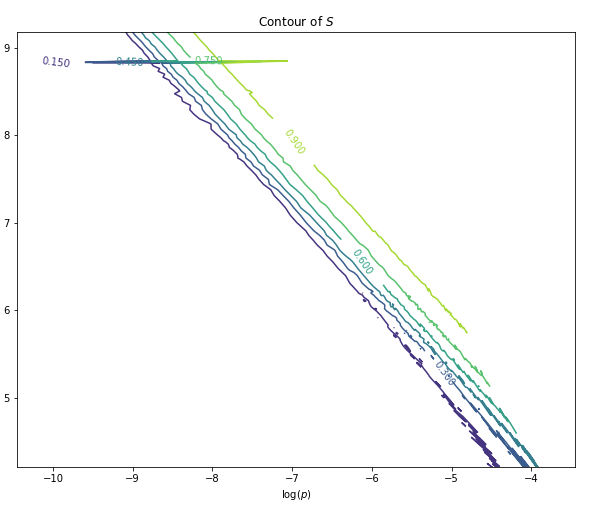


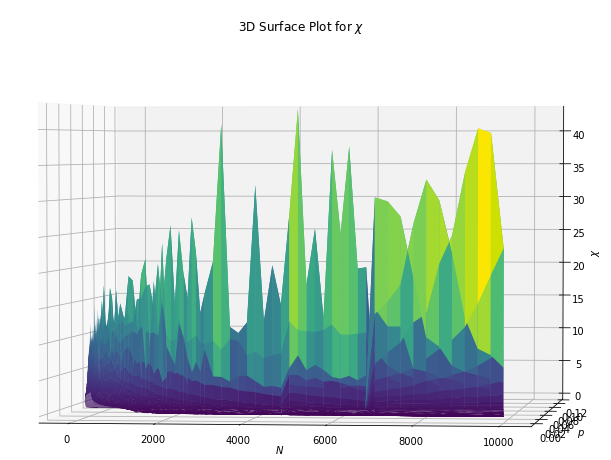
3D surface plot for *S* from above

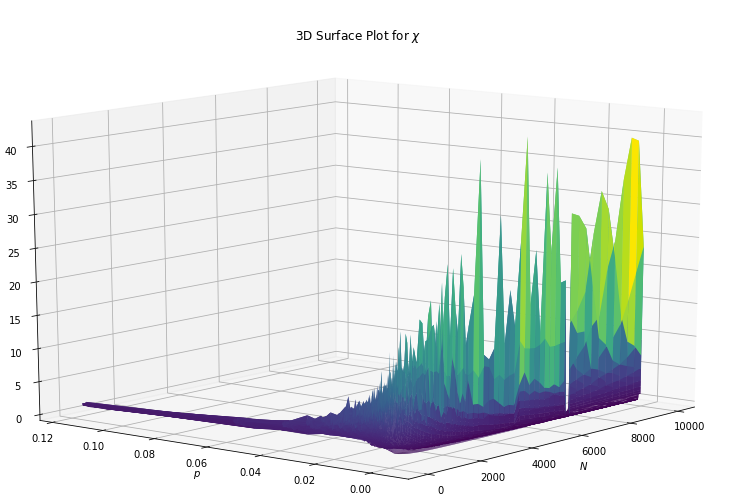
From above, we can see that the critical *pc* gets smaller as the system size increases. We also see more yellow (bigger *S*) with increasing *p* for each system size.

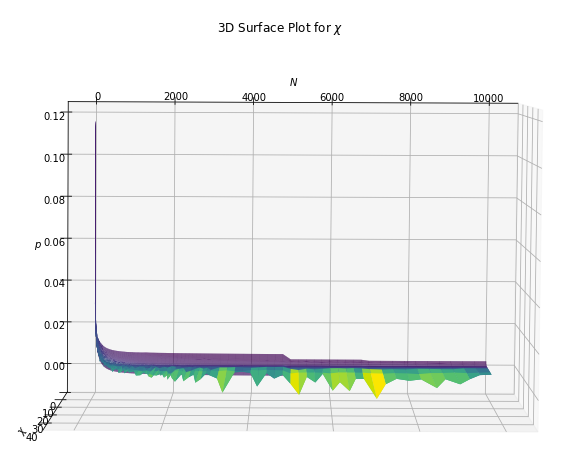


By making the contour plot for *S*, we can see the critical line clearer (the yellow line that has value 0.900).



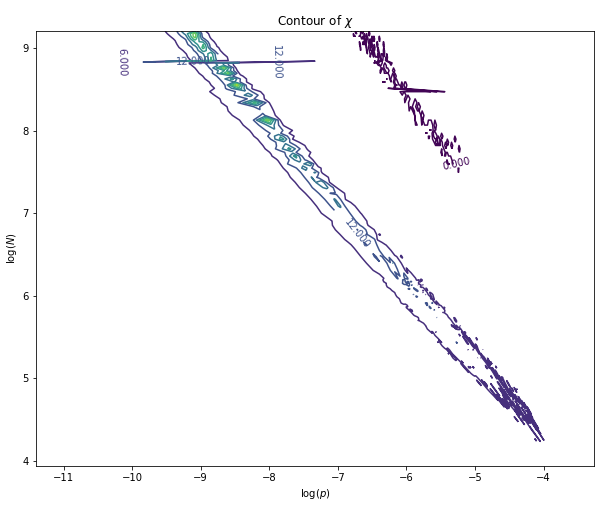
If we zoom in more, we can see that the different contours for *S* are quite well separated and seems to position according to their value (the 0.600 line is between the 0.300 line and the 0.900 line)



As for the surface plot for , one can see a confirmation of part 1 and 2. A vertical cut or a horizontal cut of this surface plot does give a multiple spike-like curve like the ones we see in the previous parts. We also see the spikes being narrower and higher for bigger systems. I am not sure why we see a gap for a certain numbers of system sizes. I think it might be caused due to the smaller neighborhood for *p* with big systems.

3D surface plot for from above

From above, we can see that the critical *pc* gets smaller as the system size increases. We can also see my sampling neighborhood gets smaller as the system size increases. We can observe that more yellow (bigger ) with increasing system size.



The contours of show a really interesting pattern. It seems like they form circles around each other. Zooming in, we can see that it seems like the contours for bigger lies within the contour for the smaller

