Activity A2 : Implementation of models of complex networks

Hajar Lachheb

1 Introduction

Through this paper, I am presenting the individual work I did. Working on the second activity was a real challenge again and a very motivating exercise indeed. It was very intriguing to read the papers and understand how each model works to be finally able to implement it.

2 Work Description

2.1 Description of what is done in all the work parts

I developed functions to generate various network models, each with different parameters, which were then used to generate networks of various sizes. These functions were implemented using Python programming language, and the networkx package was utilized to handle the creation and manipulation of the network models. The different network models generated include Barabási-Albert (BA), Erdős-Rényi (ER), Watts-Strogatz (WS), and Configuration Models.

To evaluate the generated networks, a range of parameters was used for each model. For small networks with less than 1000 nodes, I plotted a visual representation of the network using the nx.draw(G) function, and a histogram of the degree distribution. The degree distribution refers to the frequency of nodes with a particular number of edges, or degrees. For larger networks with 1000 or more nodes, I only plotted the histogram. For networks that follow a power-law distribution, the histogram was plotted in a log-log scale. Why? Because this scale is used to visualize the tail of the distribution as it is often difficult to see using a regular scale.

To estimate the exponent of the power-law distribution of degrees for networks with this property (BA and CM Scale Free), I used two methods: linear regression of log(P(k)) as a function of log(k), and Maximum Likelihood Estimation (MLE), and it is outlined in the file provided "Details-Exponent". The goal of estimating the exponent is to determine the degree to which the network follows a power-law distribution. In fact, Linear regression is a statistical technique used to identify the relationship between two variables, and in my case, it is the degree of a node (k) and its frequency (P(k)).

Following the linear regression problem, it was also necessary to discard the (b, p[b]) pairs where p[b] = 0 because it was impossible to take the logarithm of 0. For MLE, the output was often unsatisfactory. Thus, based on the article "Power-law distributions in empirical data," I figured out that it would be good to filter out all nodes with a degree less than 6 so as to improve the quality of the estimation especially when kmin = 6.

Lastly, the networks generated were saved in pajek format, along with their respective images.

2.2 Erdős-Rényi Models

ER networks are a type of random graph model that assumes that edges between nodes are formed randomly with equal probability. There are two variations of the model: G(N,K) where the graph is created with a fixed number of edges (K) and G(N,p) where the graph is created with a fixed probability (p) of having an edge between any two nodes. ER networks are useful in studying the behavior of randomly connected systems. We had to work on the Erdős-Rényi Models. I choose to work on both of them. Here are the results and plots I got.

Figure representing the ER(N p) Implementation Model

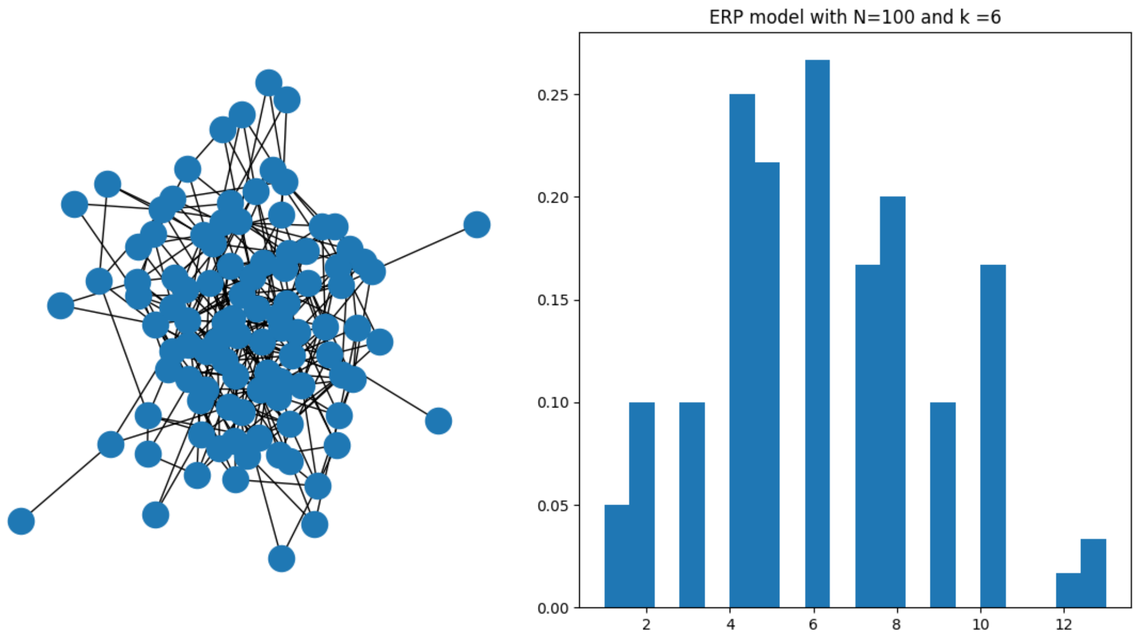
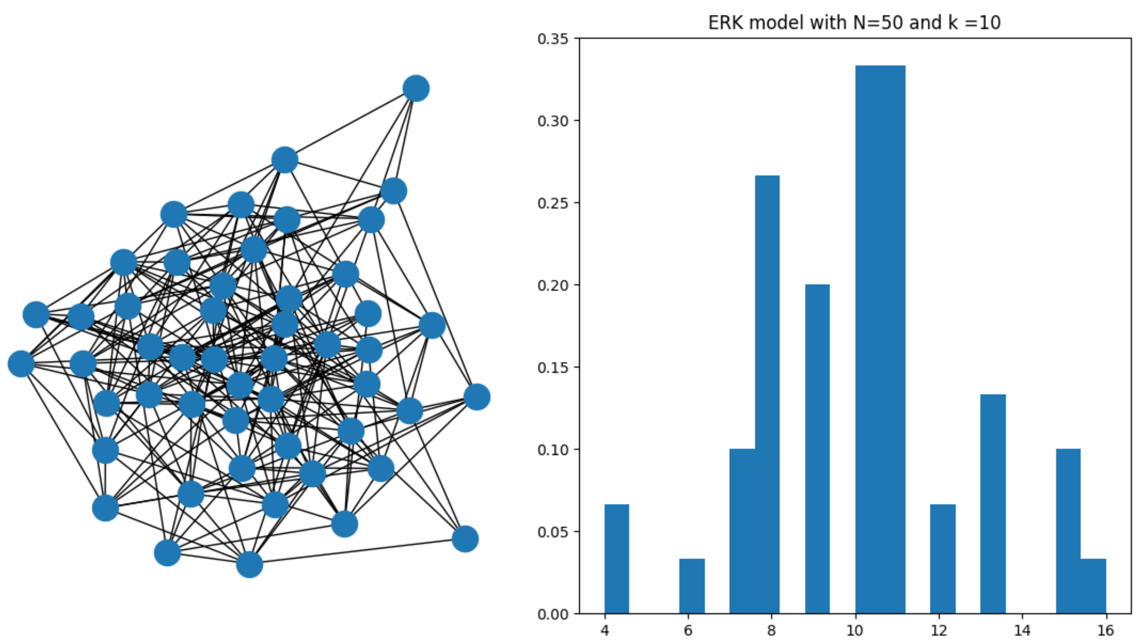


Figure representing the ER(N K) Implementation Model



My method was to plot as much plots as possible to conclude if the models with a higher N do follow or not a poisson distribution.

Figure representing the results for the ERP Model with N = 1000

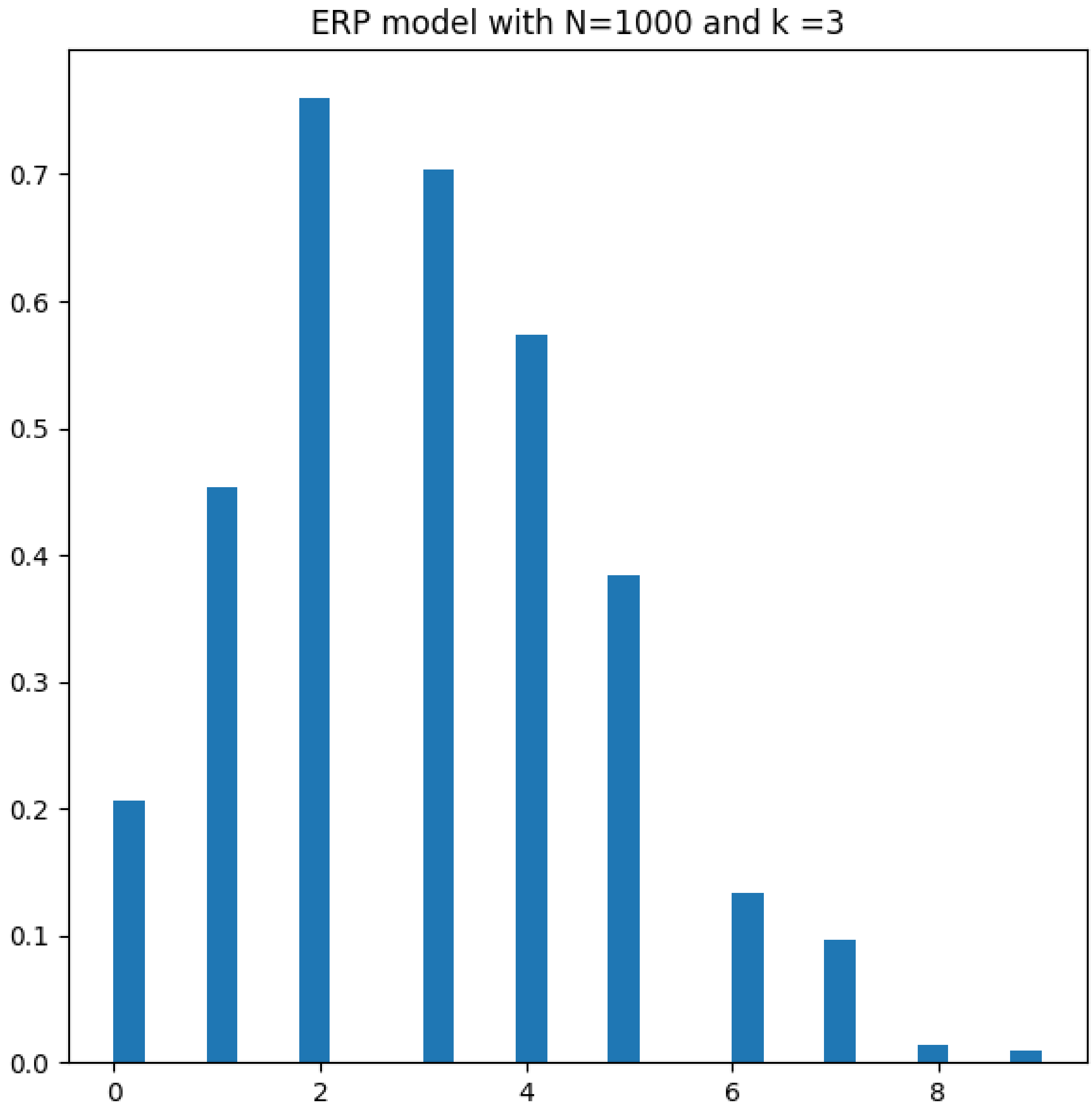
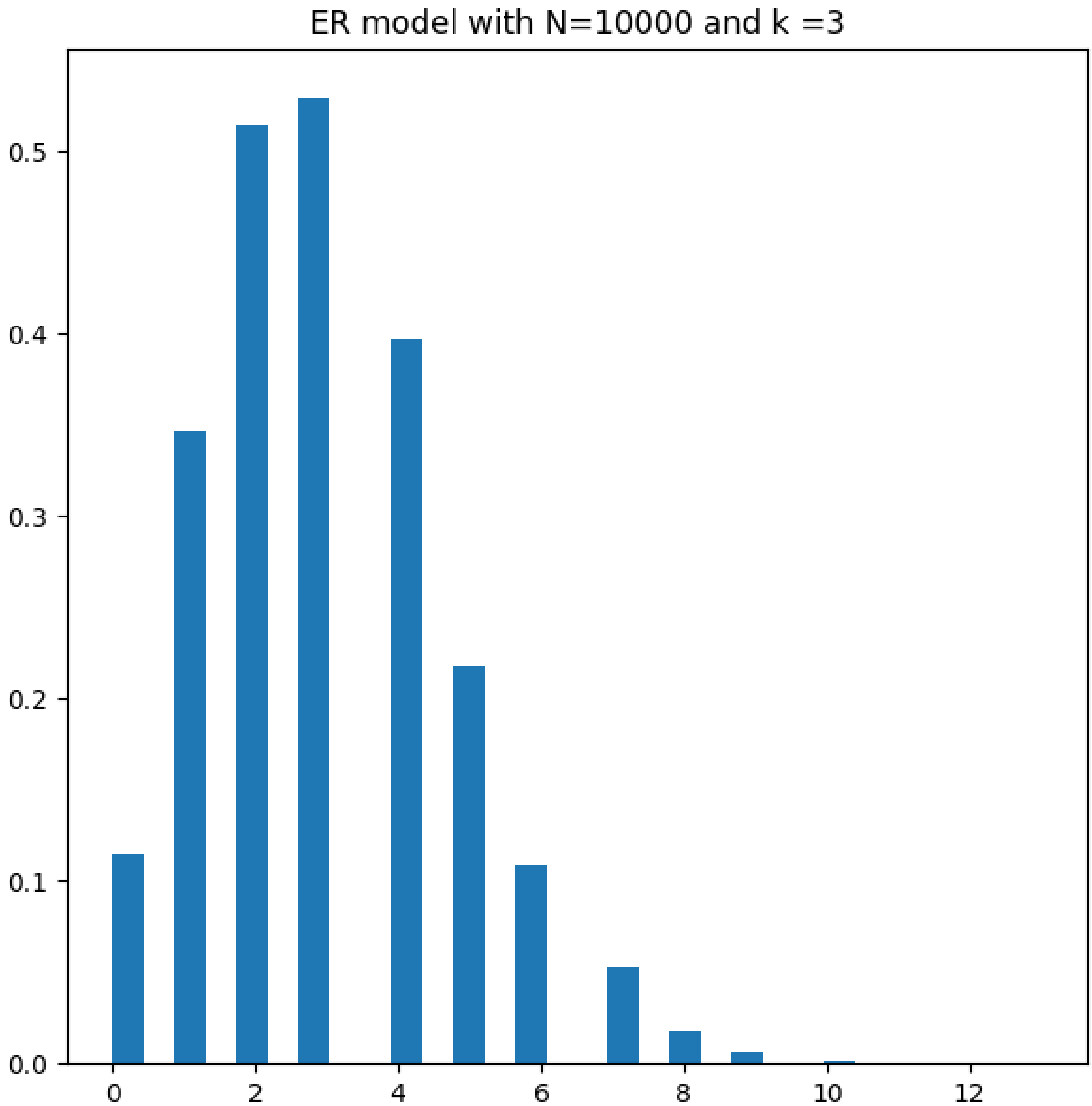


Figure representing the results for the ERP Model with N = 10000



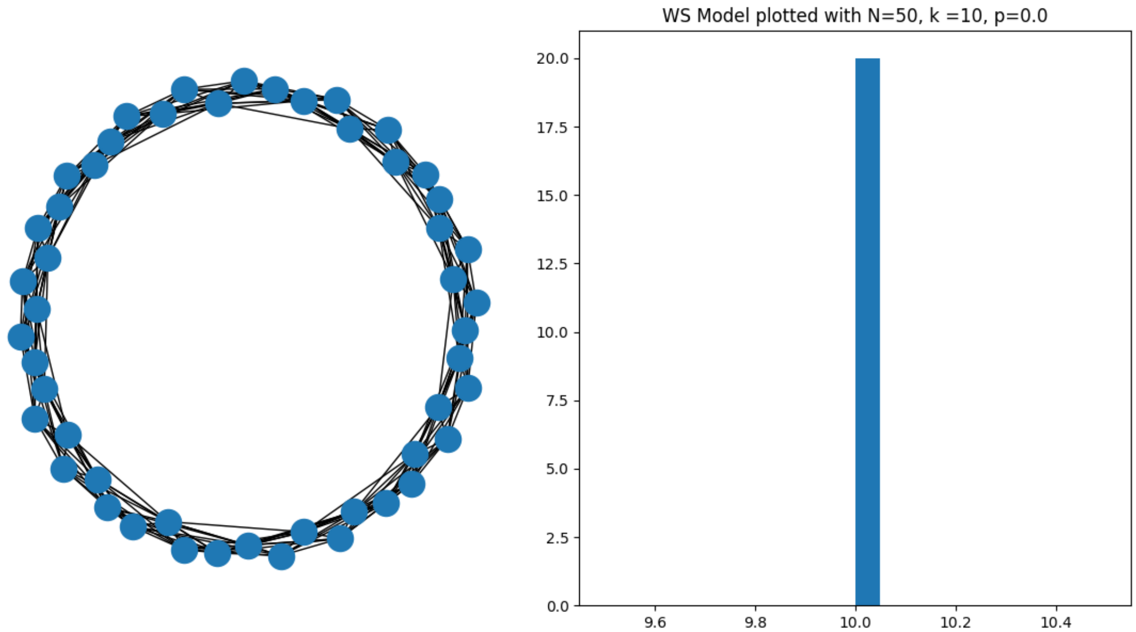
According to the theory, graphs with N greater than or equal to 1000 exhibit a Poisson distribution in their degree distribution.

2.3 Watts-Strogatz Model

The WS model is a type of network model that was designed to capture the small-world phenomenon observed in many real-world networks. It starts with a regular lattice where nodes are connected to their nearest neighbors and then rewires a certain percentage of edges randomly. This rewiring creates shortcuts that allow for efficient communication between distant nodes, while still preserving the regular structure of the original lattice.

Following we will plot the different presentation for each network and compare the results.

Figure representing the WS Model with p =0.0

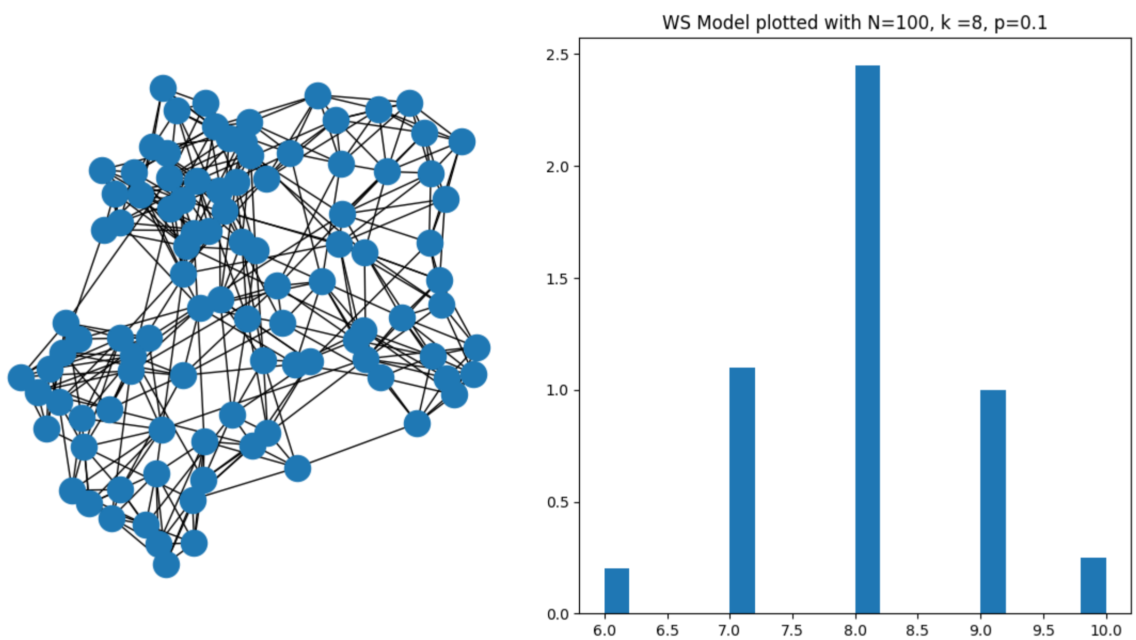


The network here is well-shaped and represents the regular ring lattice with all the nodes included and k=10

.

The resulting average path length clustering is 2.959, 0.667.

Figure representing the WS Model with p =0.1

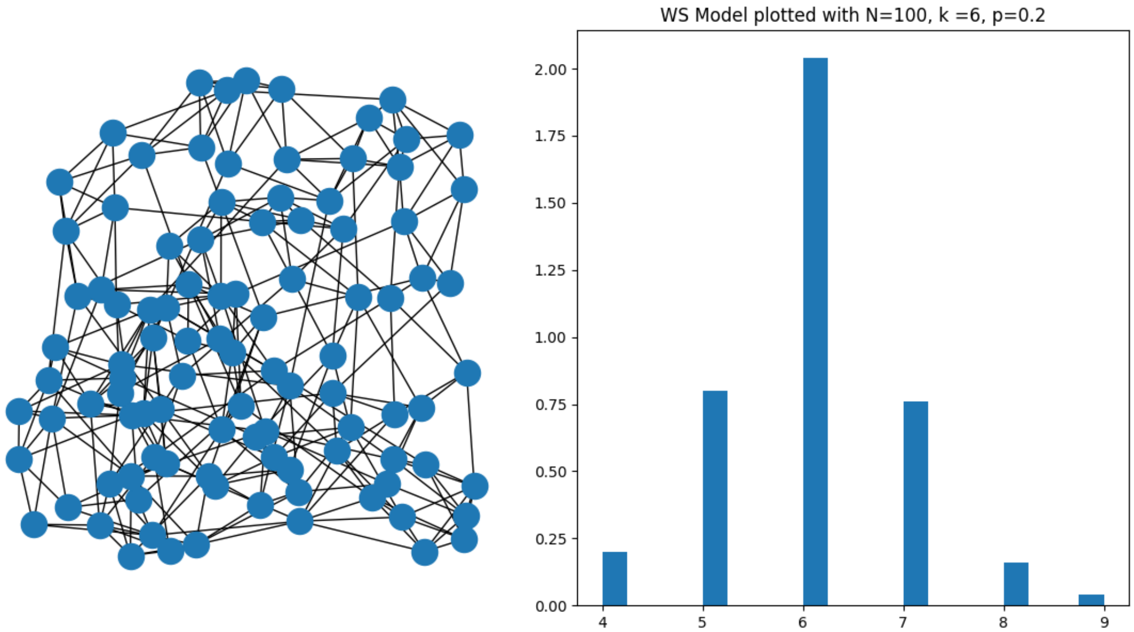


As we can see, the more p increases, the average path length, and clustering get smaller

.

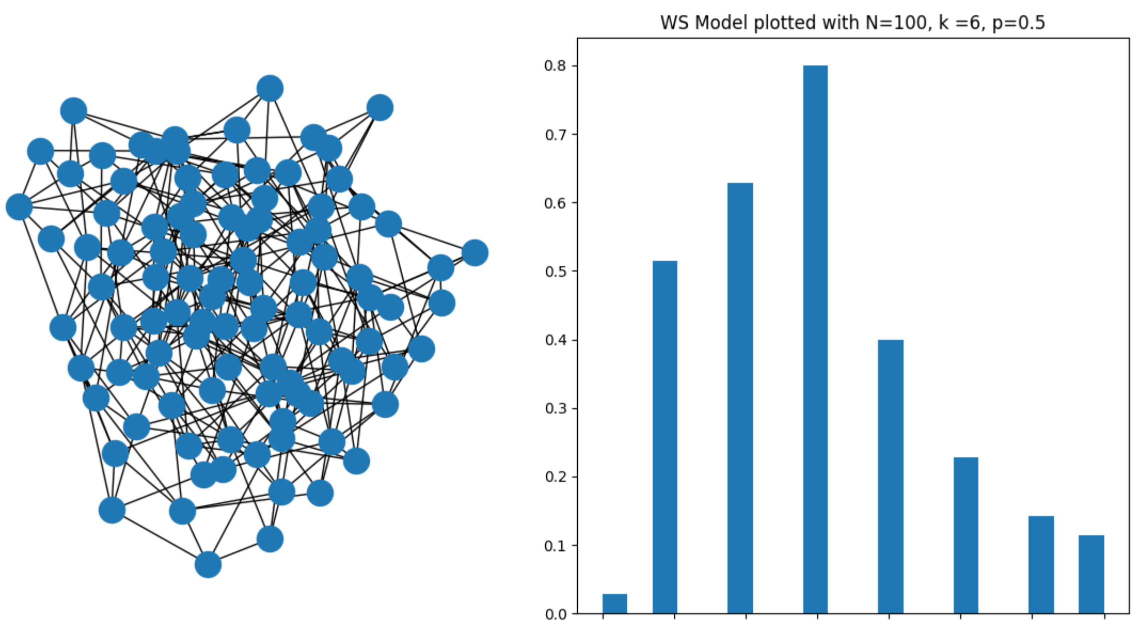
The resulting average path length clustering is 2.958, 0.455.

Figure representing the WS Model with p =0.2



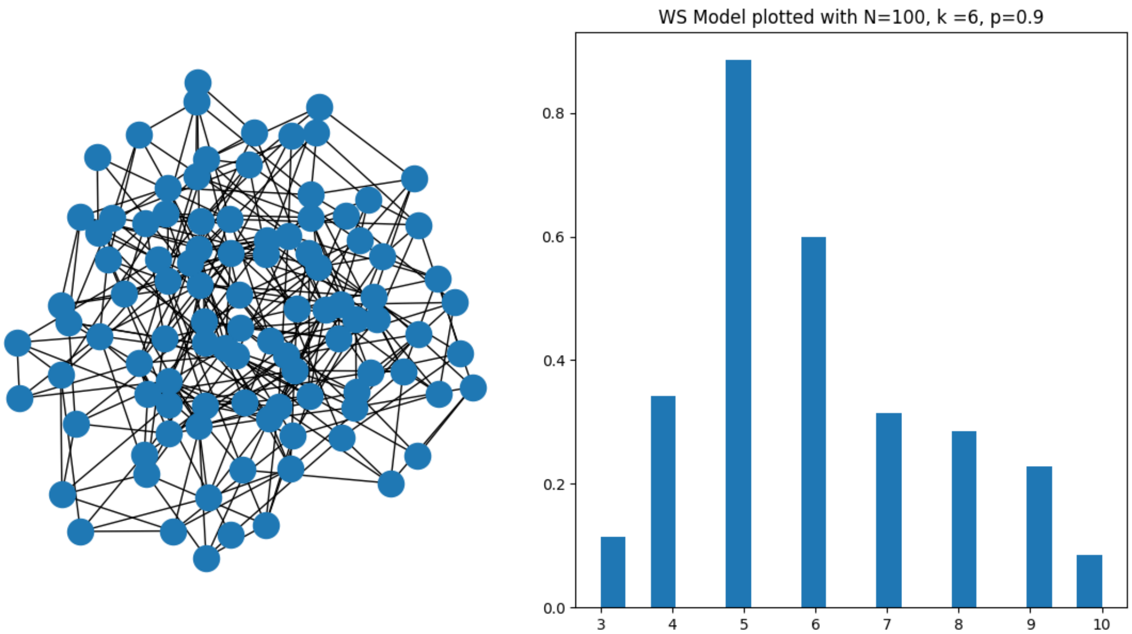
The resulting average path length clustering: 3.230, 0.339

Figure representing the WS Model with p =0.5



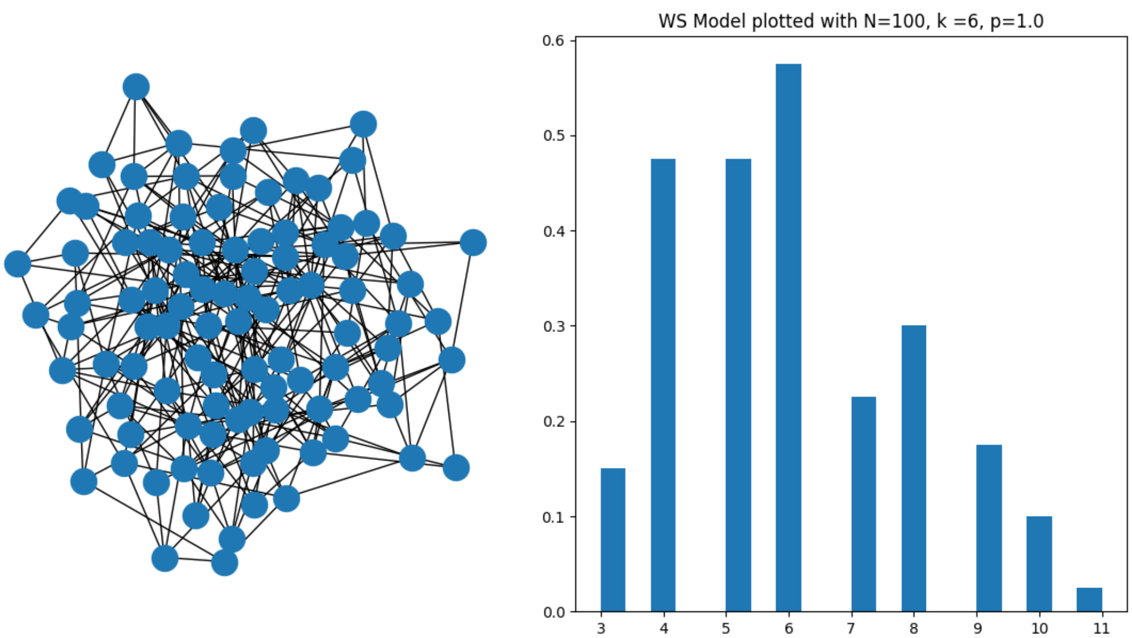
The resulting average path length clustering: 2.806, 0.108

Figure representing the WS Model with p =0.9



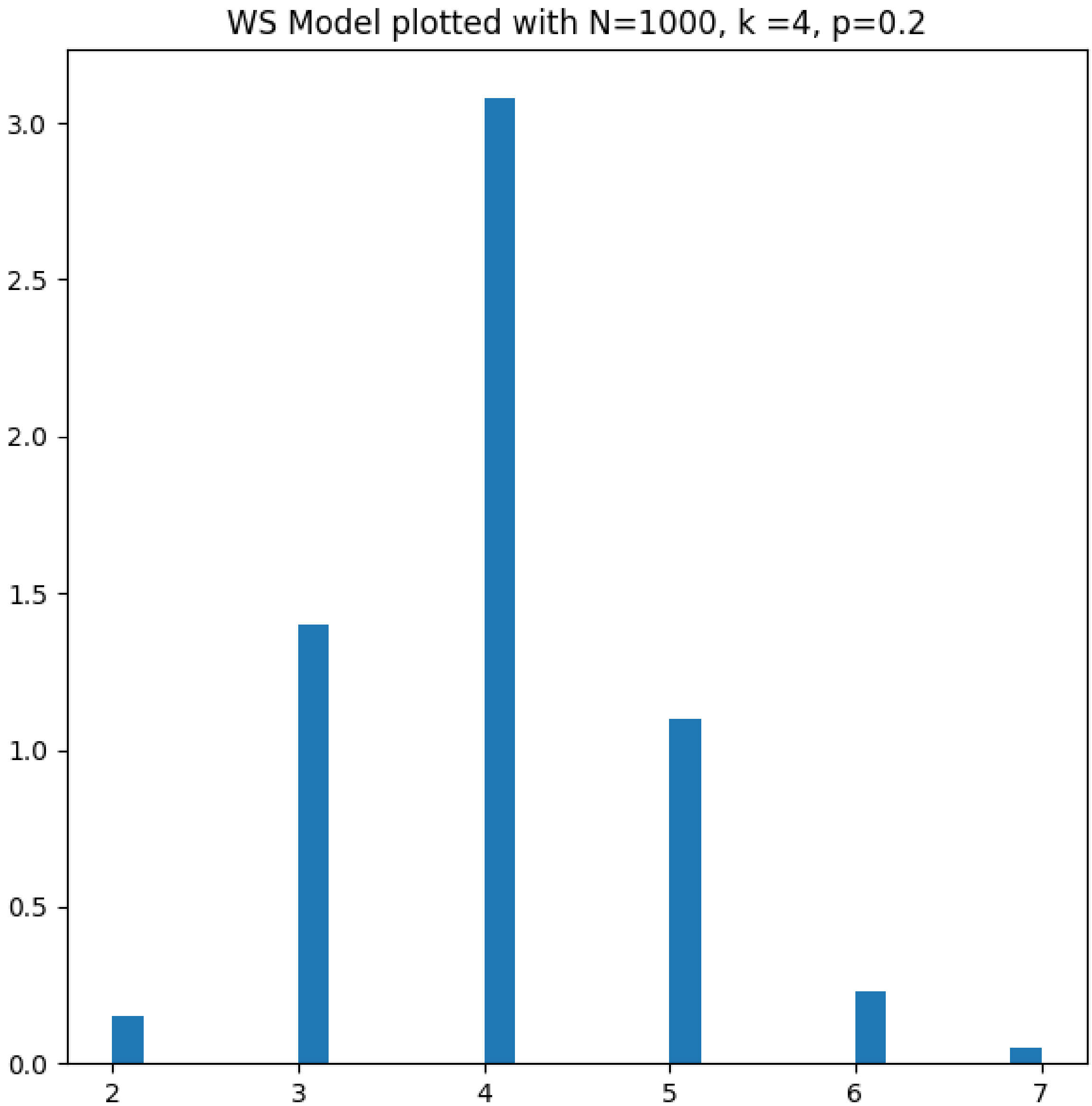
The resulting average path length clustering: 2.726, 0.053

Figure representing the WS Model with p =1.0



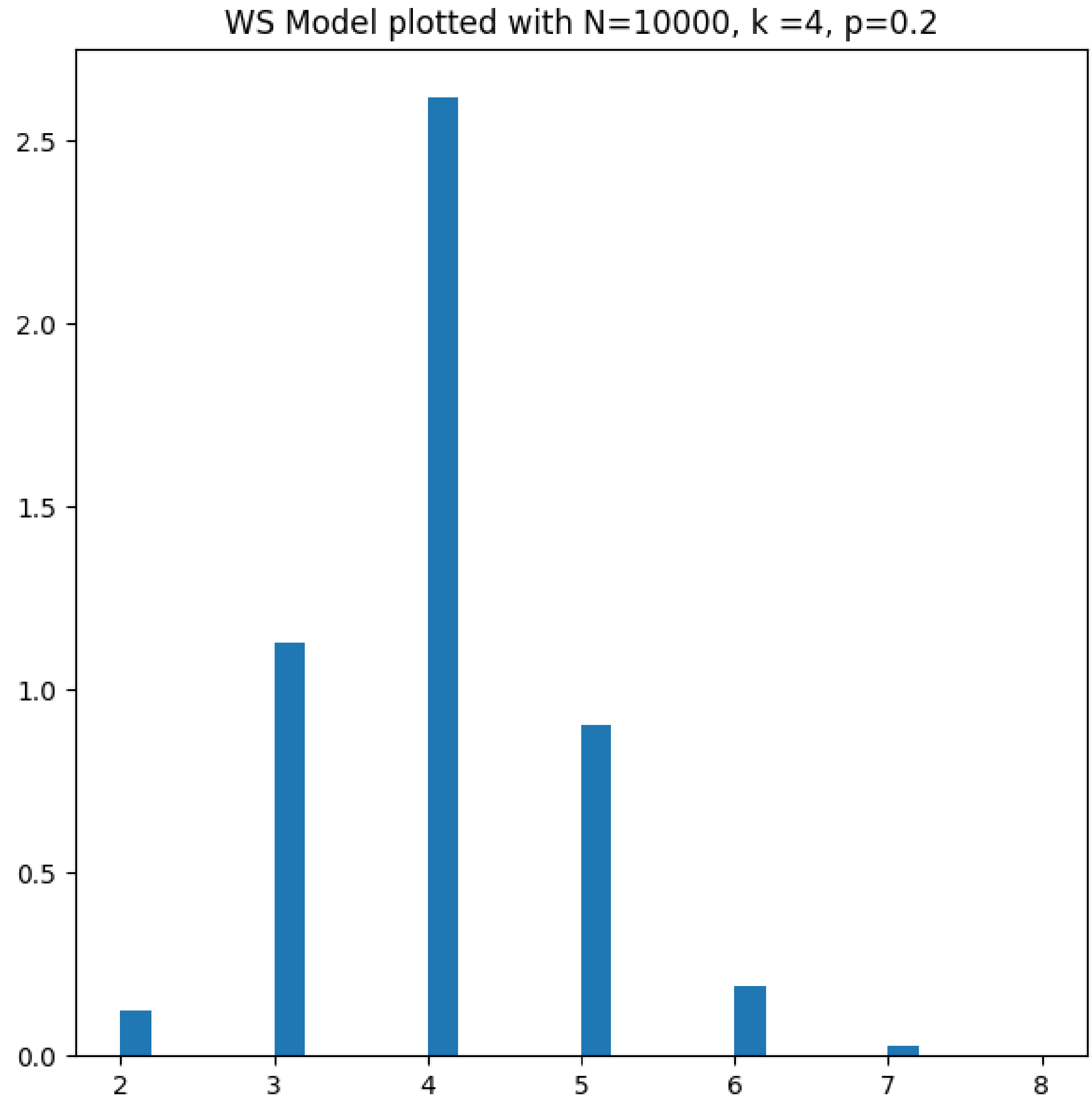
The resulting average path length clustering: 2.720, 0.058

Figure representing the WS Model with N = 1000 and p =0.2



The resulting average path length clustering: 6.876, 0.264

Figure representing the WS Model with N = 10000 and p =0.2



The resulting average path length clustering: 9.598, 0.267

2.4 Barabási Albert Preferential Attachment Model

The BA model is a type of network model that is based on the principle of preferential attachment, which means that new nodes are more likely to attach to nodes that already have a high degree. In this model, nodes are added to the network one at a time and are connected to existing nodes based on their degree. This leads to the formation of scale-free networks with a power-law degree distribution. In our case, to generate the first clique, I used the function ERNP with p=1.

Figure representing the BA Model with m =1

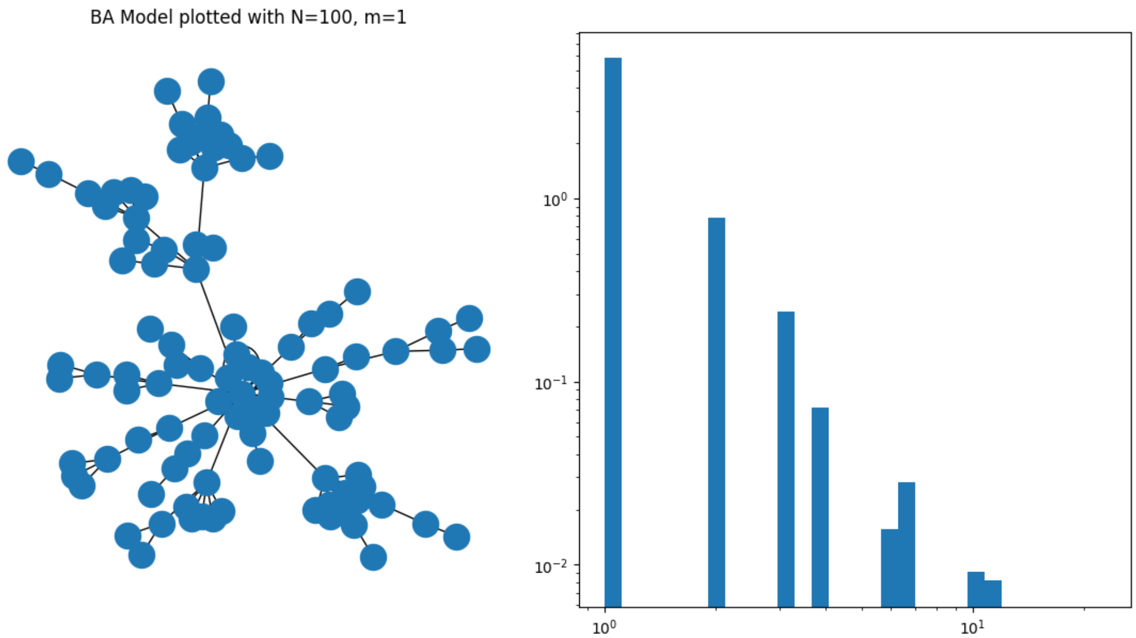


Figure representing the BA Model with m =2

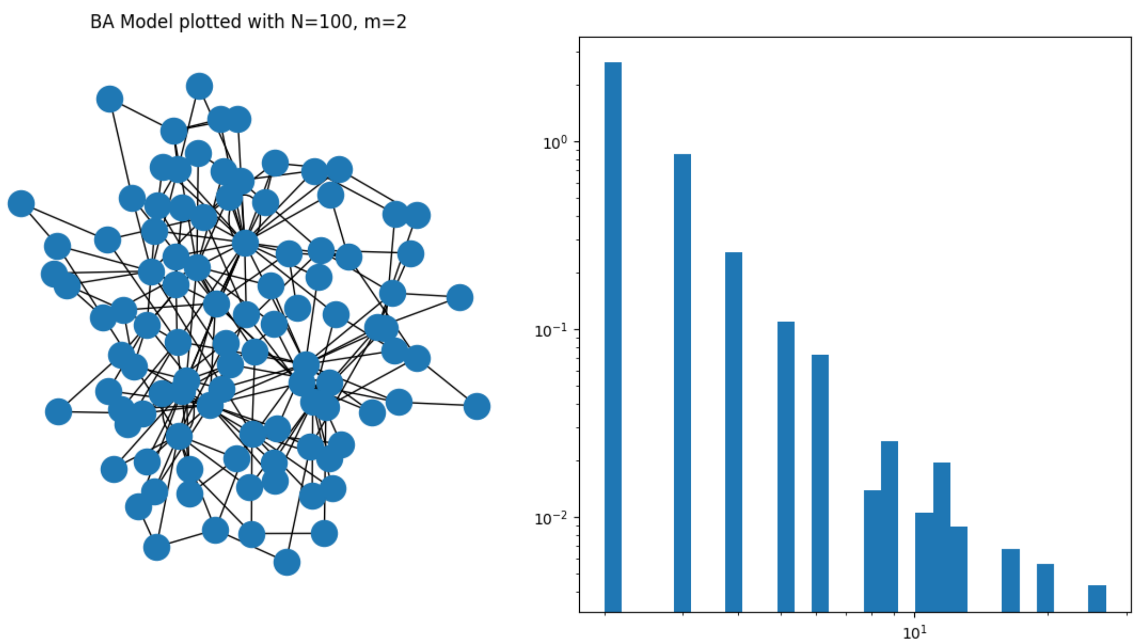
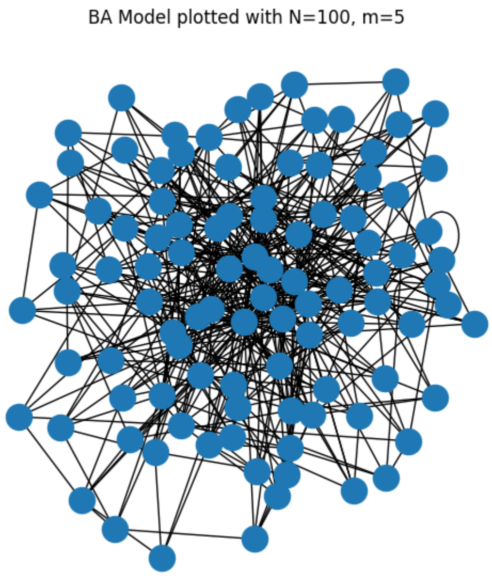


Figure representing the BA Model with m =5



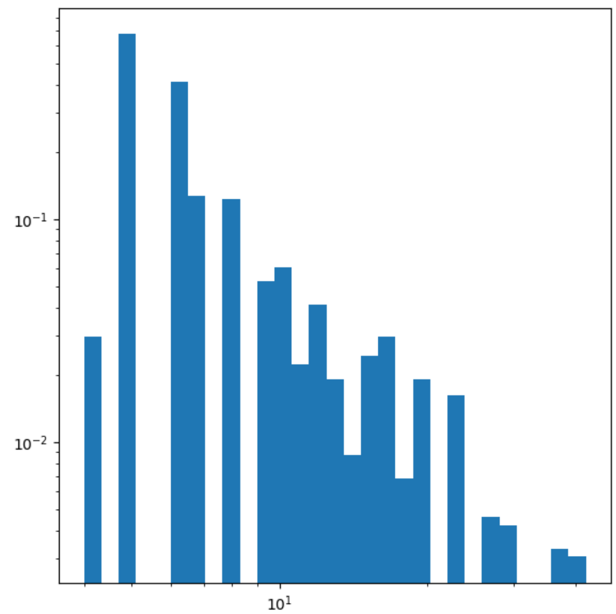
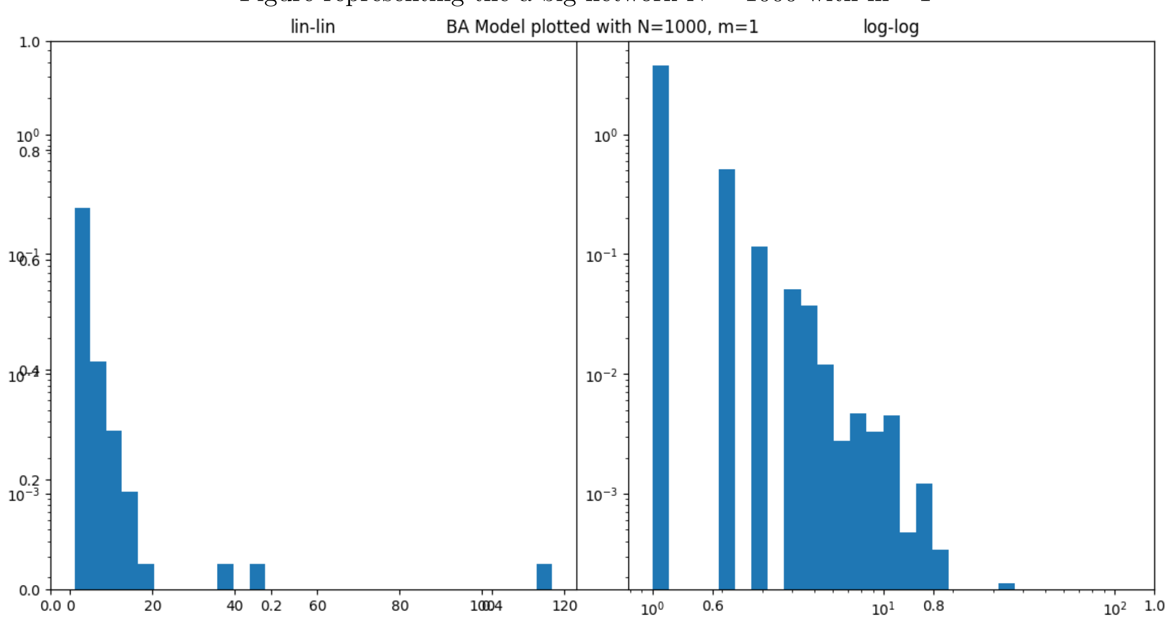
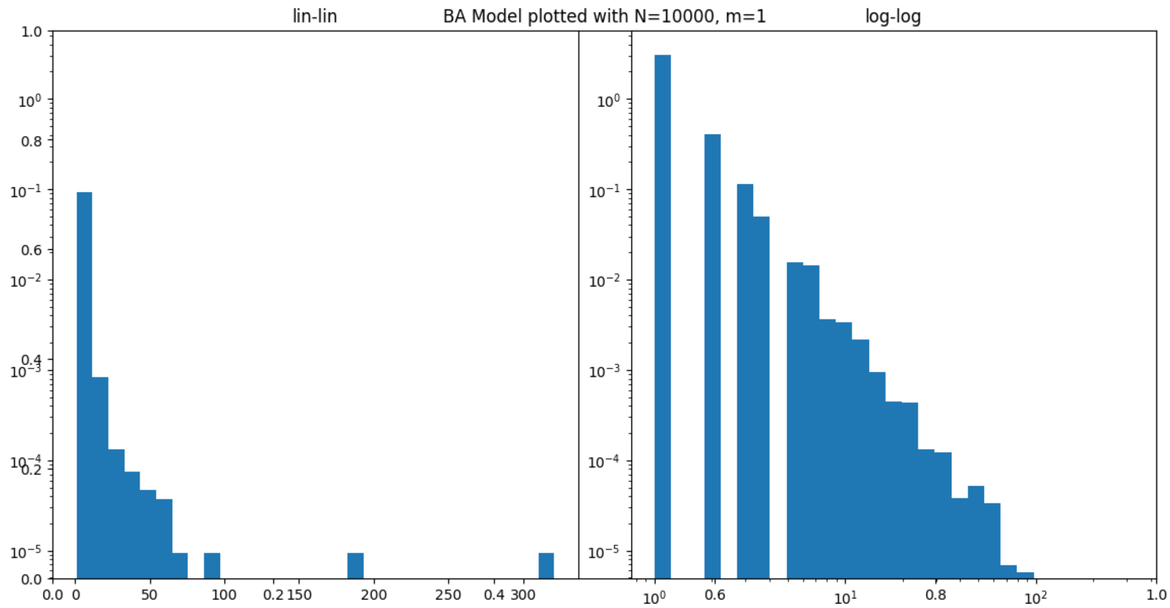


Figure representing the a big network N = 1000 with m =1



MLE gamma estimation, with kmin >= 1: -5.134536661289406 MLE gamma estimation, with kmin >5: 22.789833595369615 Regression coefficient: -1.518786907438163 , gamma: 2.5187869074381632

Figure representing the a big network N = 10000 with m =1



MLE gamma estimation, with kmin >= 1: -5.056991820408545 MLE gamma estimation, with kmin >5: 26.148303180830407 Regression coefficient: -1.6557410373594104 , gamma: 2.6557410373594106

2.5 Configuration Model

The CM is a network model that generates random graphs with a specified degree distribution. It starts by creating a set of "stubs" for each node in the network, where the number of stubs is equal to the degree of the node. The model then randomly pairs up the stubs to form edges, subject to the constraint that each stub can only be paired up once. The result is a random graph with a degree distribution that matches the specified distribution of the original network. The CM is often used to study the impact of degree distributions on network behavior. In our case and after some research, we will use the scale-free distribution as well as the poisson one.

Figure representing the CM Model with a Poisson Distribution

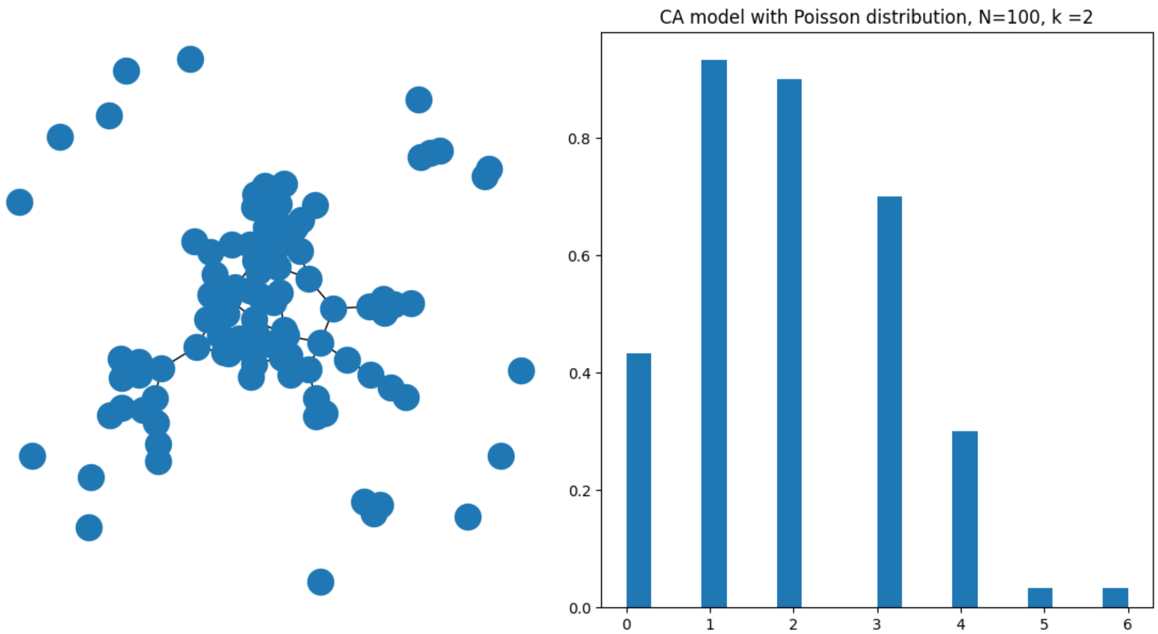


Figure representing the big networks, N = 1000 and k = 4

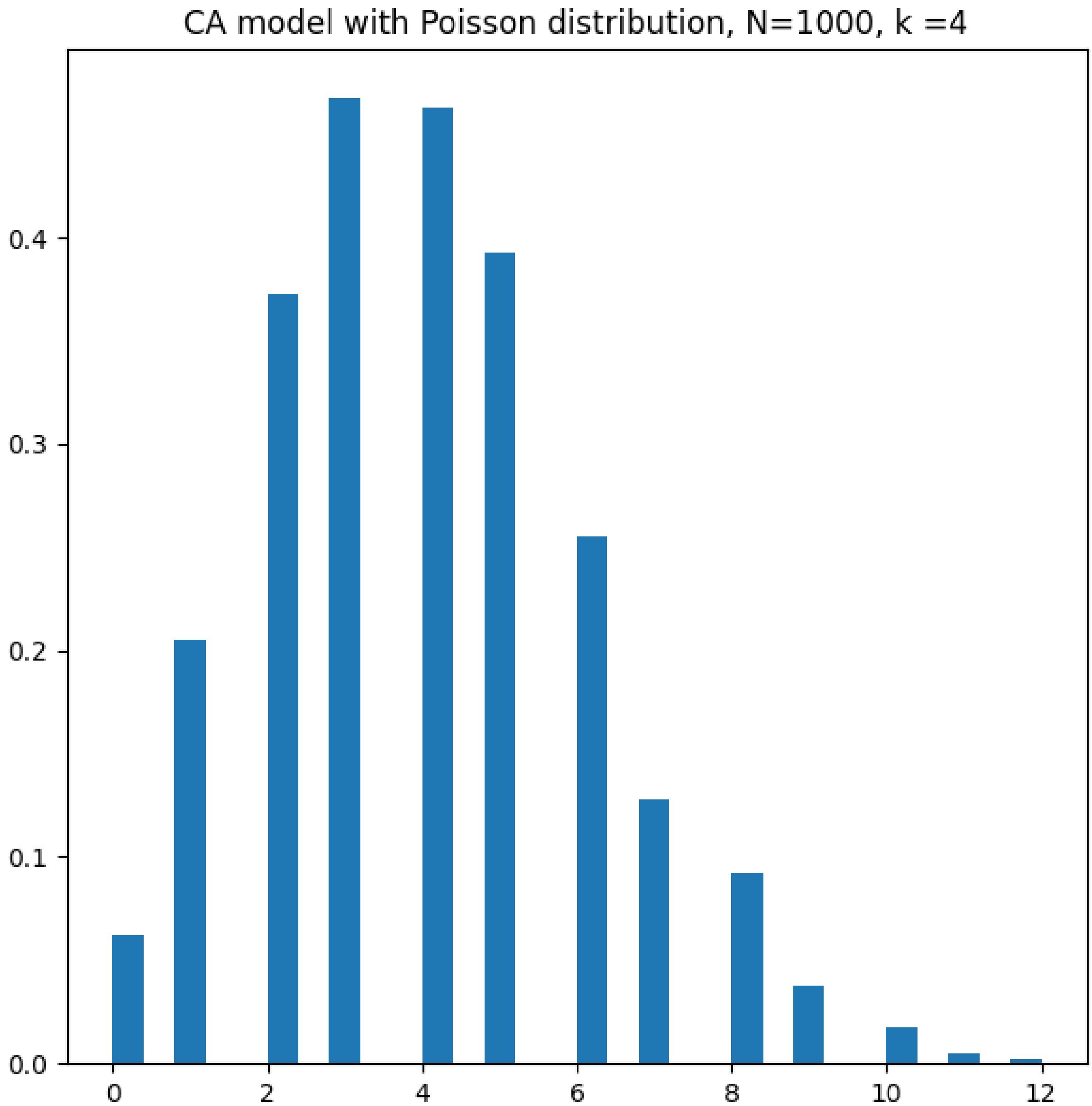


Figure representing the big networks, N = 10000 and k = 4

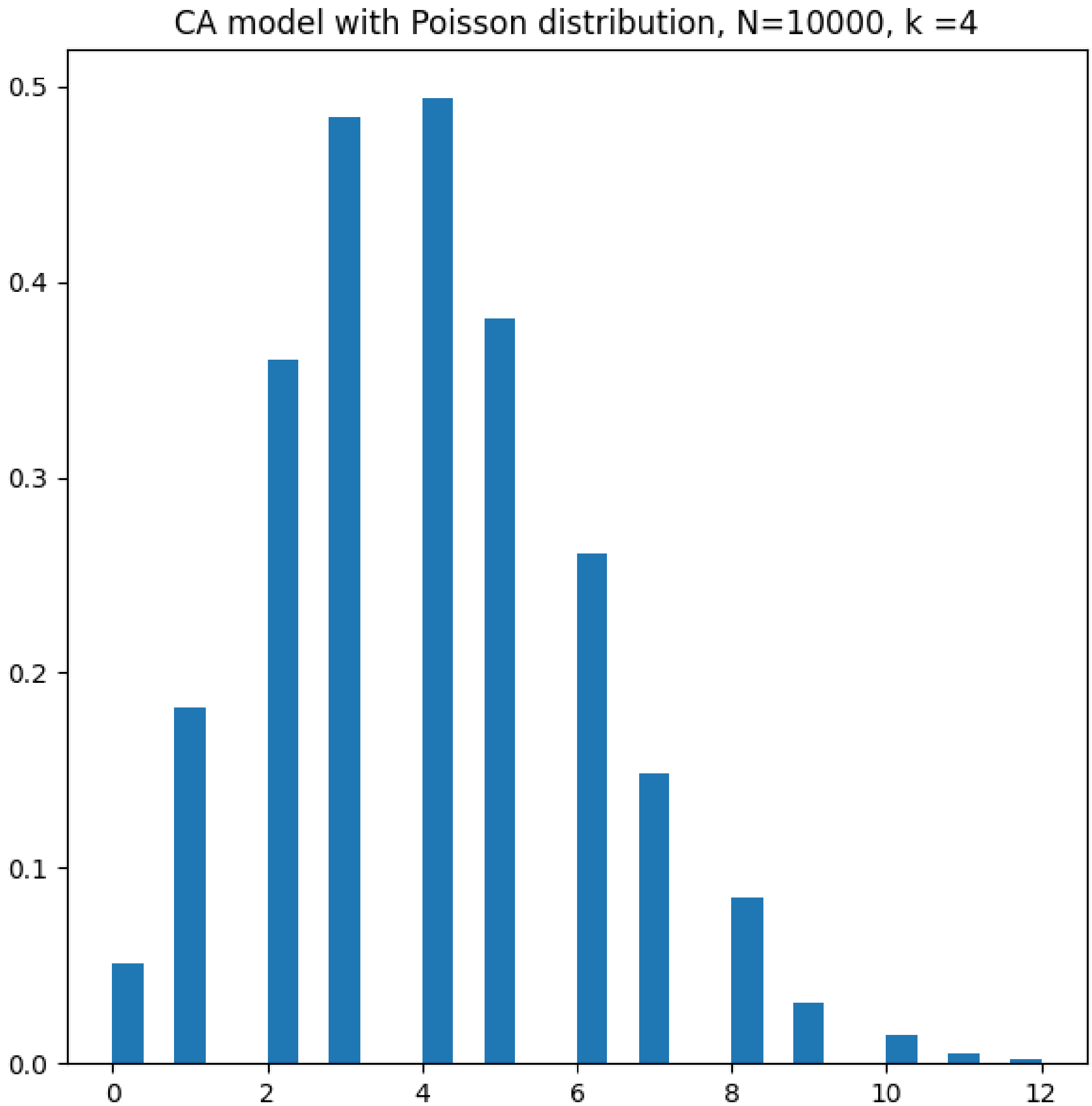


Figure representing the CM with a Scale Free Distribution

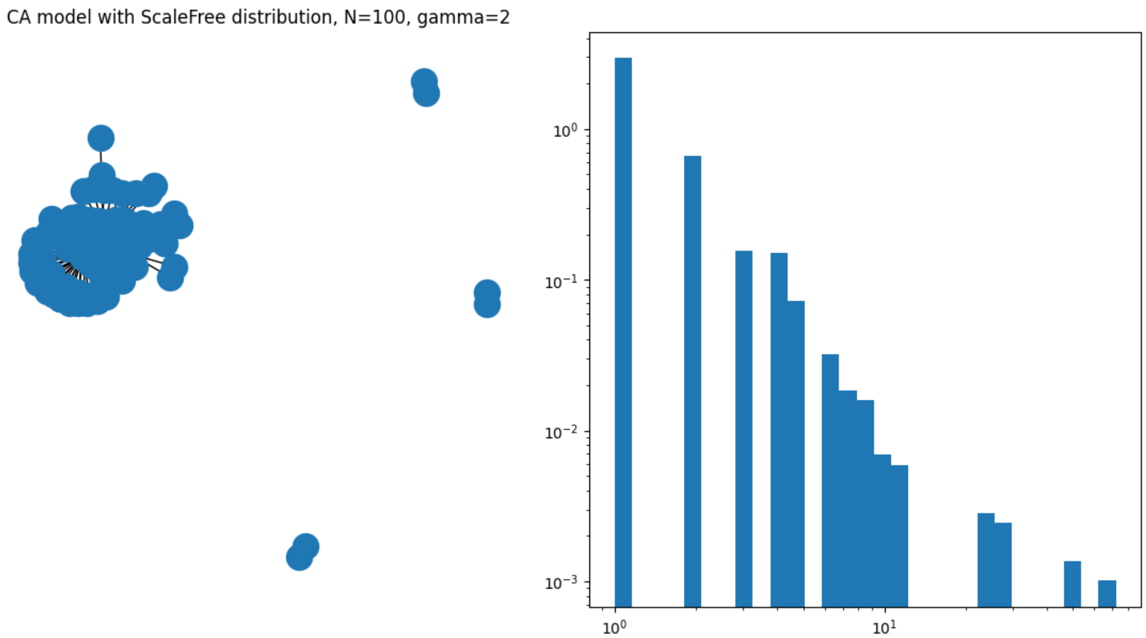
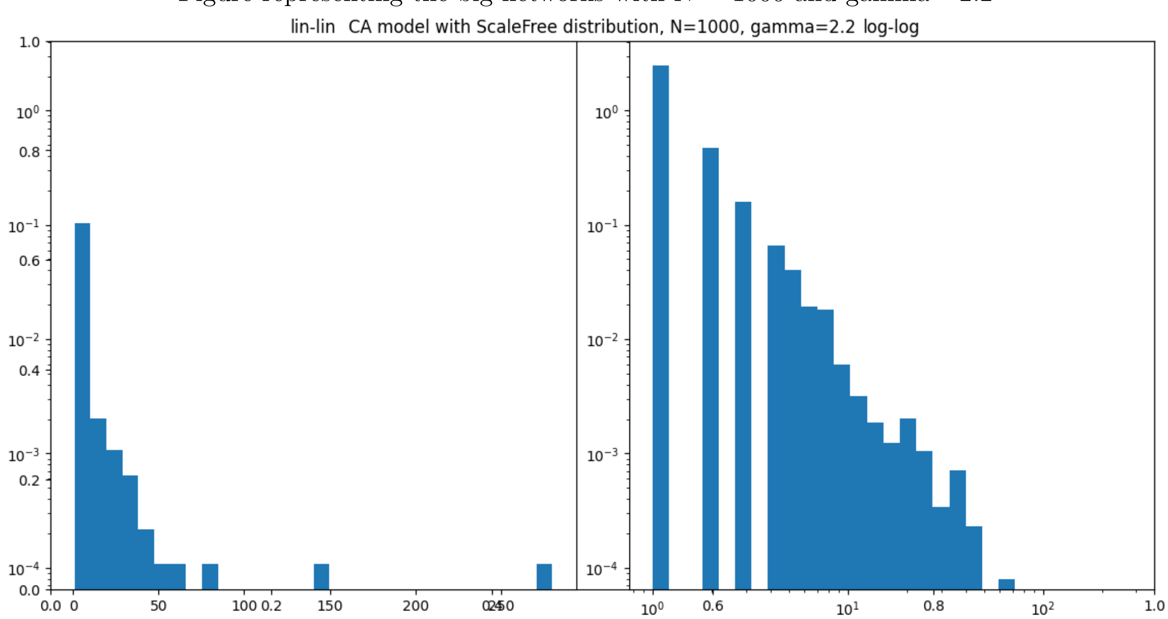
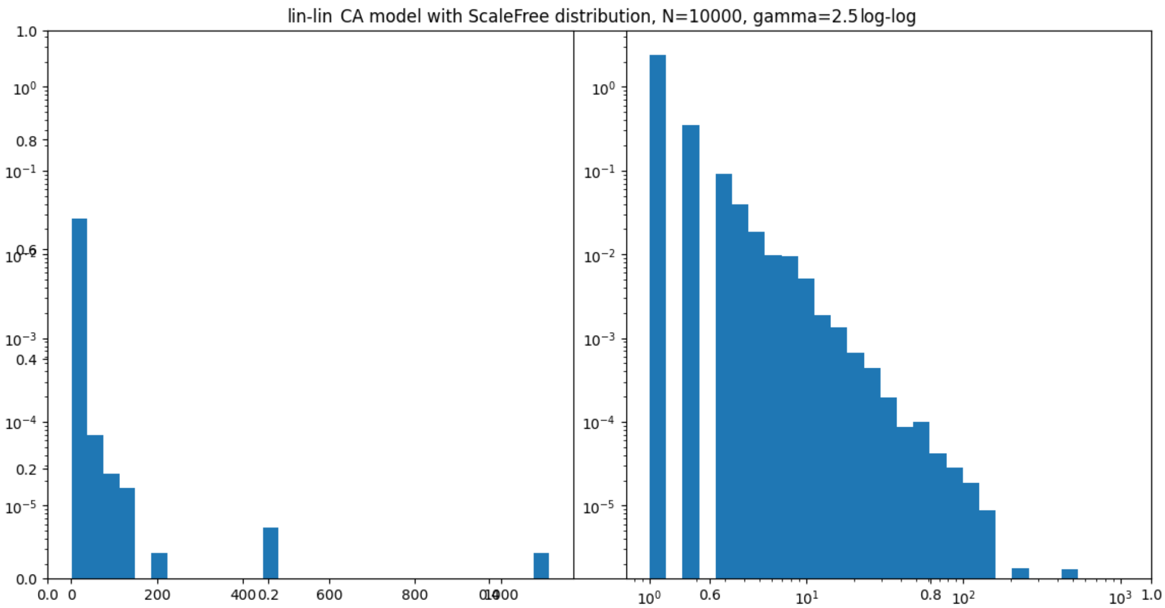


Figure representing the big networks with N = 1000 and gamma =2.2



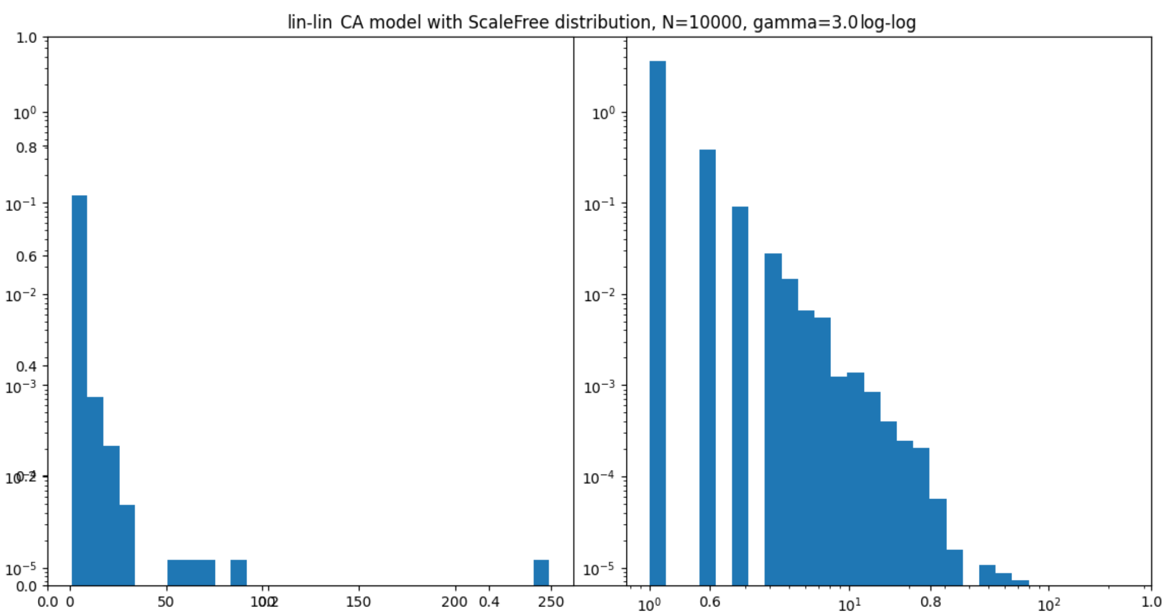
gamma: 2.2 MLE gamma estimation, with kmin >= 1: 6.984923012512734 MLE gamma estimation, with kmin >5: 5.090084942686498 Regression coefficient: -1.2224734675221713 , gamma: 2.222473467522171

Figure representing the big networks with N = 10000 and gamma =2.5



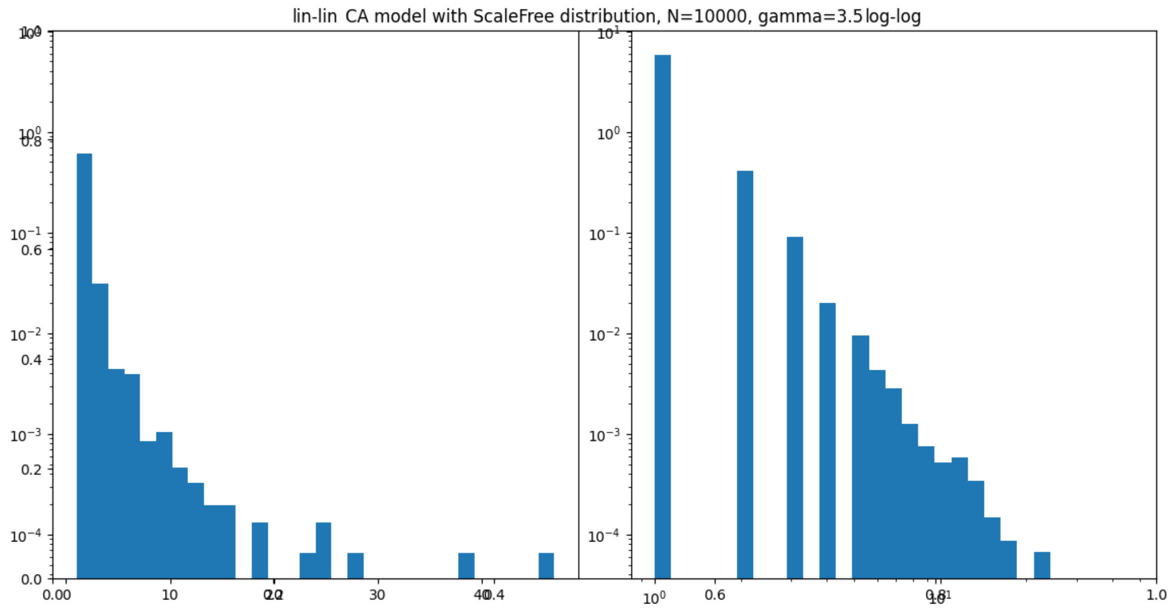
gamma: 2.5 MLE gamma estimation, with kmin >= 1: -9.911864487232965 MLE gamma estimation, with kmin >5: 5.633951018509934 Regression coefficient: -1.2823242293033015 , gamma: 2.2823242293033017

Figure representing the big networks with N = 10000 and gamma =3.0



gamma: 3 MLE gamma estimation, with kmin >= 1: -2.215665004161247 MLE gamma estimation, with kmin >5: -16.933978295467067 Regression coefficient: -1.8462359674753752 , gamma: 2.8462359674753754

Figure representing the big networks with N = 10000 and gamma =3.5



gamma: 3.5 MLE gamma estimation, with kmin >= 1: -1.260094110224554 MLE gamma estimation, with kmin >5: -6.092911094913975 Regression coefficient: -2.462858912945493 , gamma: 3.462858912945493