Complex Networks
Assignment 2

Denaldo Lapi and Samy Chouiti

April 4, 2022

Contents

1 Software and Decisions

We decided to use Python and the "NetworkX" module [1] in order to use their graphs and nodes data structures. However, we implemented by ourselves the different network model generators.

The generated networks can be found in the *nets* folder, but please note that networks with a number of nodes N higher than 1000 were not included.

In this report we will provide only the most meaningful plots and analysis. For all the detailed steps of our algorithm's implementation, we suggest to look at the provided Python code; that can be found as described below.

1.1 Structure of the assignment folder

The assignment folder contains several files and folders described here:

- *generators.py*: Contains the network generators for ER and WS.
- *test_generators.py*: Contains simple tests of the generators ER and WS.
- *network_utils.py*: Contains plotting and fitting methods that we built (on this assignment or the previous one), including PDF and CCDF with Poisson and Binomial models.
- *WS_model.ipynb, ER_model.ipynb, CM_model.ipynb and BA_model.ipynb*: Notebooks used for the analysis and network generators in case of BA and CM.
- *nets/*: Contains generated network in Pajek format.

[1] Networkx documentation

2 Erdos-Renyi model (ER)

For this model, we chose to implement the classic Erdos-Renyi model and not the Gilbert's variant. This model takes the following parameters:

- N: Number of nodes
- K: Number of edges

and is based on the following principle: **each edge connects a randomly selected pair of nodes that is not already connected.**

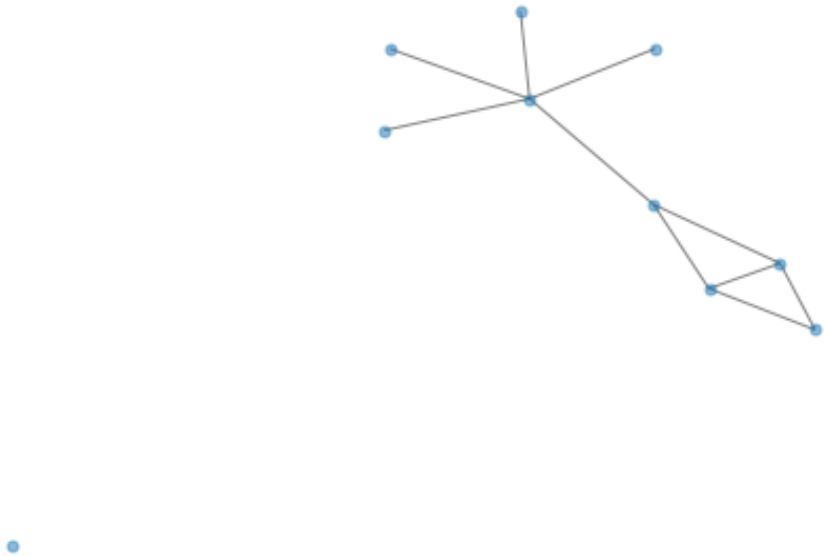

Figure 1: An ER Network G($N = 10$, $K = 10$)

This network can be found under the name *ER_N10_K10.net* in the *nets* folder.

2.1 Generation of the network

For this network, we simply created a graph with N nodes then add K nodes between randomly selected nodes (which are not creating a self-loop and that are not already connected).
The code used for generation can be found in the *generators.py* with the method *ErdosRenyi(N, K)*.

2.2 Overview of generated networks

In the following part, we generated several networks in order to make observations on the networks generation, depending on its parameters. As N being a trivial parameter, we will demonstrate the influence of K on a fixed number of nodes (here $N = 50$). Because the assignment required us to use specific values of the mean degree $< K >$, we have first to apply the following formula for our network generation: $K = \frac{N<k>}{2}$.

$< K >$	K
3	75
6	150
9	225

Table 1: N and $< K >$ values for $N = 50$

Therefore, here is the comparison for $< K >$ and K:

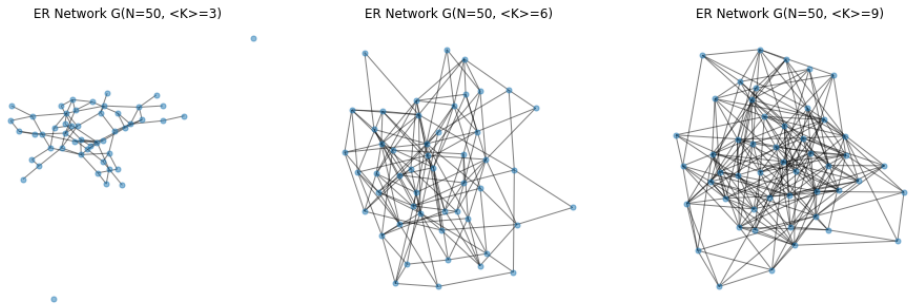

Figure 2: ER: G($N = 50$, $K = 3$), G($N = 50$, $K = 6$), G($N = 50$, $K = 9$)

The first plot demonstrates the example of having a too small K number which lead to the creation of single-node components. Also, as expected, we can see that the higher K is, the more dense the network is. Those 3 networks can be found under the names *ER_N50_mK3.net*, *ER_N50_mK6.net* and *ER_N50_mK9.net* saved in Pajek format, under the *nets* folder.

2.3 Degree distributions and modelisation

In this part, we will make observations about the experimental degree probability distribution and confirm the theoretical modelisation.
Because of the fundamental property of an ER network to randomly creating edges between nodes, its probability distribution can be modeled by a binomial law:

$$P(k) = \binom{N}{k} p^k (1-p)^{k-1}$$

with $p = \frac{2K}{N(N-1)}$ when using ER classical version (compared to Gilbert's variant).
Also, for high N values, the binomial law can be approximated by a Poisson law of parameter $\lambda = Np$, which is the model we are going to compare our networks to.

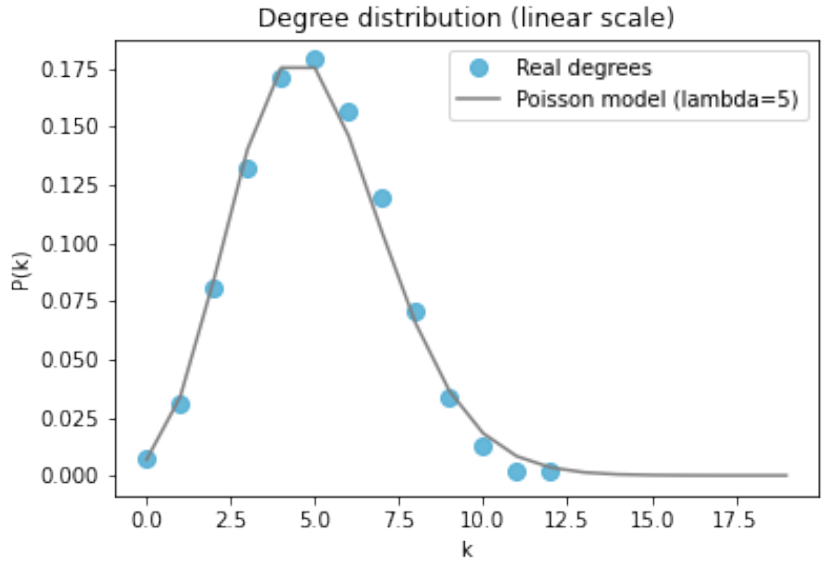

Figure 3: Degree Distribution of a G($N = 10^3$, $K = 5$) (linear scale)

We can see that for $N = 10^3$, we can already moodel the probability distribution by a Poisson law of the parameter $\lambda =< k >$.

2.4 Parameter estimations

For this model, the estimation of the parameter K is pretty straightforward because of the construction property of the network. The Maximum Likelihood Estimator (MLE) of a Poisson law is:

$$\hat{\lambda} = \frac{1}{N} \sum_{i=1}^{N} k_i$$

with k_i being the degree of the i-th node, which is by definition of $< k >$:

$$< k >= \hat{\lambda}$$

3 Watts-Strogatz (WS)

This model takes for parameters:

N: Number of nodes

K: Mean degree (even number)

p: Rewiring probability (detailed below)
and its is based on a regular network with random rewiring (based on the p parameter). It can be used to observe the Small-World property.

3.1 Generation of the network

To generate a WS network, we first create a regular ring lattice of N nodes with each node being connected to its k nearest neighbors (thus $\frac{k}{2}$ neighbors on clockwise and counter-clockwise side). In our algorithm, we iterate over each node to add the edges with the $\frac{k}{2}$ neighbors on the clockwise side which adds each needed edges at the end.
Then we apply the **rewiring property**: for each node, rewire each clockwise edge with probability p to another node, without creating self-loops or multi-edges.

3.2 Overview of generated networks

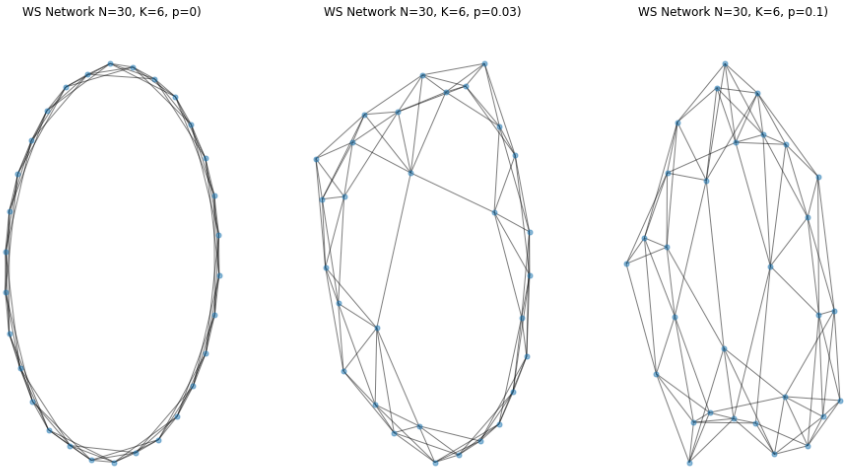

Figure 4: WS: $N = 30$, $K = 6$ and $p \in \{0, 0.03, 0.1\}$

On the different graphs above, we can see the strong influence of the p parameter. When $p = 0$, there is no rewiring applied, thus the network stays in its initial state which is a regular ring lattice. However, even for a small value of p such as $p = 0.03$, we can see the rewiring influence which is even greater with $p = 0.1$: multiple initial edges are removed and new ones are created,

thus notably changing the structure.

Those 3 networks can be found under the names *WS_N30_K6_p0.net*, *WS_N30_K6_p0.03.net* and *WS_N30_K6_p0.1.net* saved in Pajek format, under the *nets* folder.

3.3 In depth analysis of the rewiring probability influence

Visual modifications aside, we can also observe the influence the rewiring probability on two descriptors of the network:

- Clustering coefficient C
- Average shortest path length L

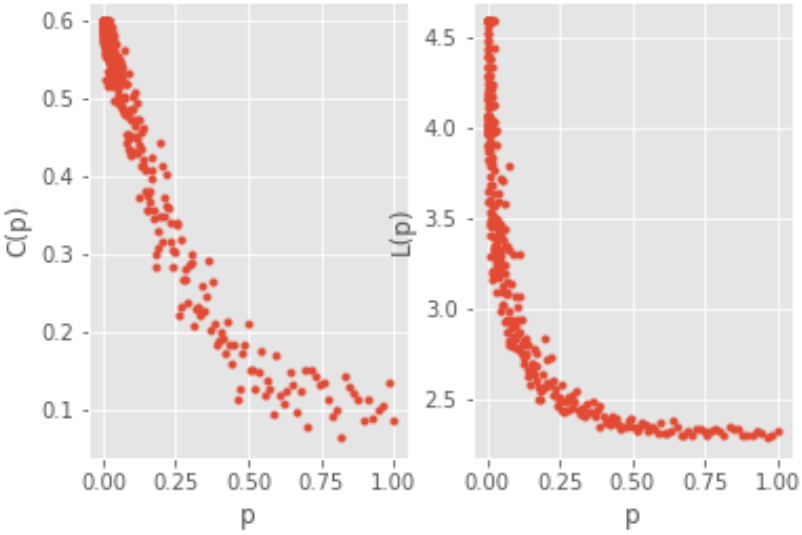

Figure 5: WS Clustering coefficient and Average shortest path length for $N = 50$, $K = 6$ and $p \in [10^{-4}, 1]$ (on a logspace)

Even though we can observe that both curves are converging, we will normalise their values using $C(p)/C(0)$ and $L(p)/L(p)$ in order to plot both on the same graph:

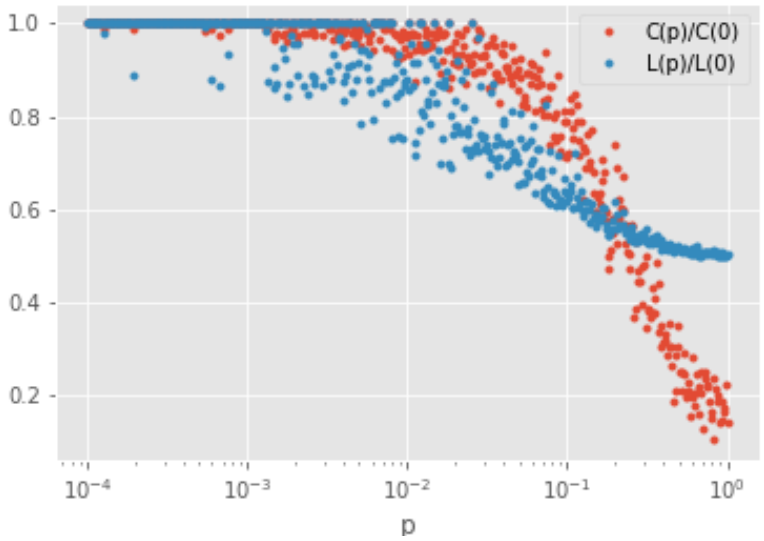

Figure 6: WS Both normalised Clustering coefficient and Average shortest path length for $N = 50$, $K = 6$ and $p \in [10^{-4}, 1]$ on a logarithmic x-axis scale

On this common plot, we can see that:

- The normalized average shortest path length $L(p)/L(0)$ seems to converge toward 0.5, meaning that the higher the rewiring probability is, the more likely we are increasing by a factor of two the average shortest path length between two nodes.
- The clustering coefficient is decreasing when the rewiring probability is higher but don't seem to converge. The only conclusion we can draw here is that the rewiring process is highly removing clusters, compared to a network without rewiring.

3.4 Degree distributions and modelisation

As can be found in the "On the properties of small-world network models" paper [2], the degree distribution of a WS generated network can be approximated by a Dirac model centred on the K value.

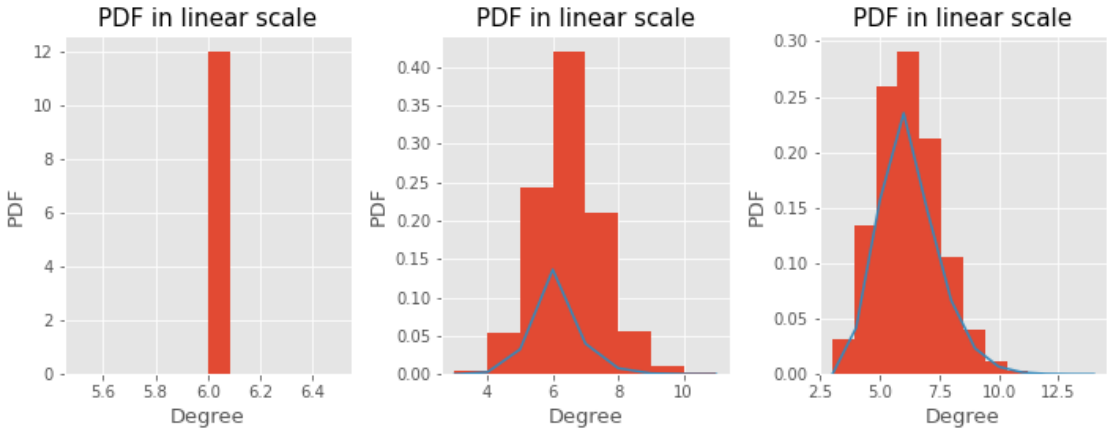

Figure 7: WS PDF for $N = 10^4$, $K = 6$ and $p = 0$, $p = 0.2$, $p = 0.5$ (left to right).
Real distribution in red, Dirac distribution in blue.

On the left plot, we can see that all degrees of the network are equal to $K = 6$ as the rewiring probability is 0. Therefore, we observe the expected degree distribution of a regular ring lattice.

On the two other graph, we are able to see the influence of the rewiring probability as the the peak degree is more emphasized with $p = 0.2$ than with $p = 0.5$. It makes sense as the more rewiring we apply to the network, the more the degree distribution is going to be flattened.

[2] Barrat, A.; Weigt, M. (2000). "On the properties of small-world network models". European Physical Journal B. 13 (3): 547–560. arXiv:cond-mat/9903411

4 Barabási–Albert model (BA)

The BA model generates random free-scale networks (i.e. with a power-law degree distribution) using a preferential attachment mechanism.

It is based on two main principles:

- Network growth: the size of the network is increased starting from an initial number of nodes by incrementally adding new nodes each one with a fixed number of edges.
- Preferential attachment: new nodes are connected to the existing nodes according to the idea of preferential attachment, i.e. more connected nodes have more probability to "attract" new nodes.

4.1 Generation of the network

The detailed steps of our implementation of the algorithm are showed in the notebook *BA_model.ipynb*.

We just briefly explain here the main inputs to the BA algorithm, needed in order to run the corresponding notebook:

- *n_0*: number of initial nodes
- N: number of final nodes
- m: number of links for each new added node (should be lower than m_0)

By changing these 3 parameters, the algorithm will build a network accordingly to the chosen values.

An important note is that the Python implementation of the algorithm, by using Lists data structures and the Networkx functions is very heavy: the time required by the algorithm to build the model with 10.000 nodes made the initial generations not feasible. In order to improve the performance of the algorithm, we decided to use the *Cython* extension [3], which allow to use C static compiling inside the Python code (all the details are shown in the notebook). By using this trick we were able to obtain a great improvement in the performance of the algorithm, which is now able to build a network with 10.000 nodes in a very few seconds.

4.2 Overview of the generated networks

During our experiments we generated networks of different sizes, but let's see first the plots of some of the small-size networks we generated.

In particular, we analyzed the influence of the m parameter on the structure of the network. The figure 8 demonstrates that the higher m is, the more dense the network is. Indeed the 2 networks have the same size N and same *n_0*. Those 2 networks can be found under the names *BA_n04_m1_n100.net*, *BA_n04_m3_n100.net* , under the *nets* folder.

In figure 9 we report 2 networks of size 50. We can easily draw the same conclusions.

Those 2 networks can be found under the names *BA_n02_m2_n50.net*, *BA_n05_m3_n50.net* , under the *nets* folder.

[3] Cython documentation

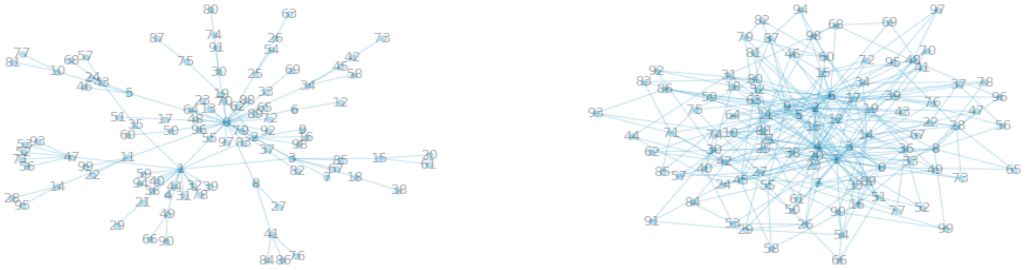

Figure 8: BA networks with n_0 set to 4, N set to 100 and m equal to 1 on the left and 3 on the right

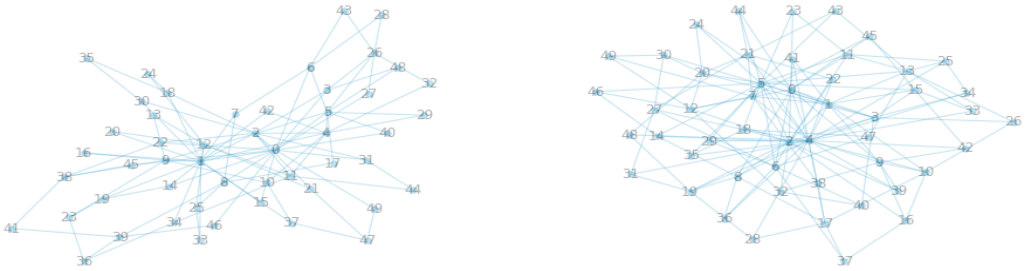

Figure 9: BA networks of size 50. On the left: n_0 set to 2, m equal to 2. On the right: n_0 set to 5, m equal to 3

4.3 Degree distributions and modelisation

In this part, we will make observations about the experimental degree probability distribution and confirm the theoretical modelisation.

We know that the degree distribution of Barabási–Albert network is K^{-3}, this means that the BA model generates scale-free networks and so it gives a straight line in log-log scale. Let's visualize some of these power-law distributions in the figures 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

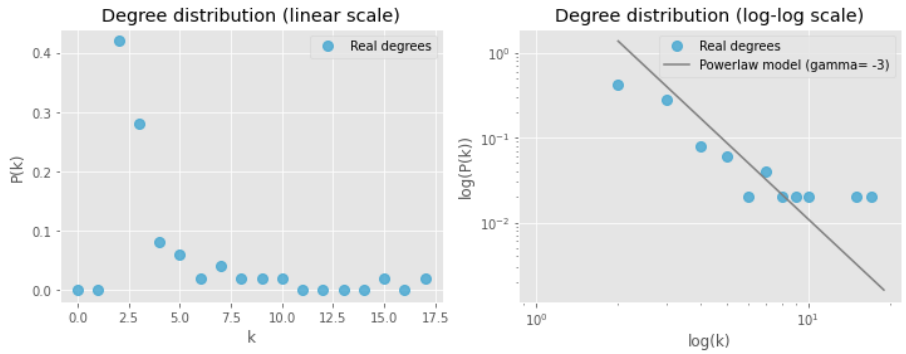

Figure 10: BA with parameters: n=50, n_0=2, m=2

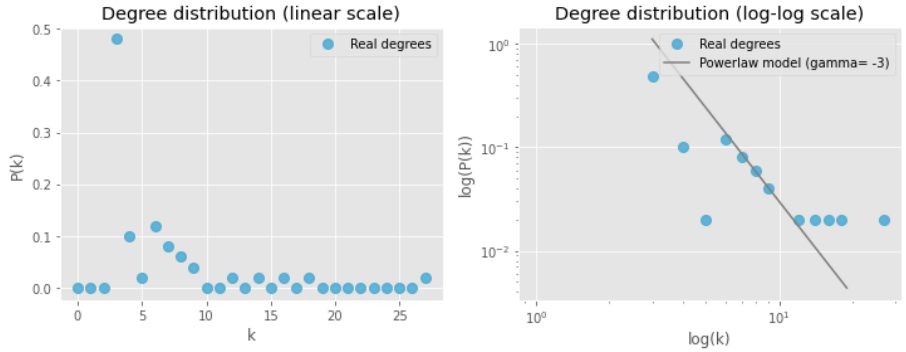

Figure 11: BA with parameters: n=50, n_0=5, m=3

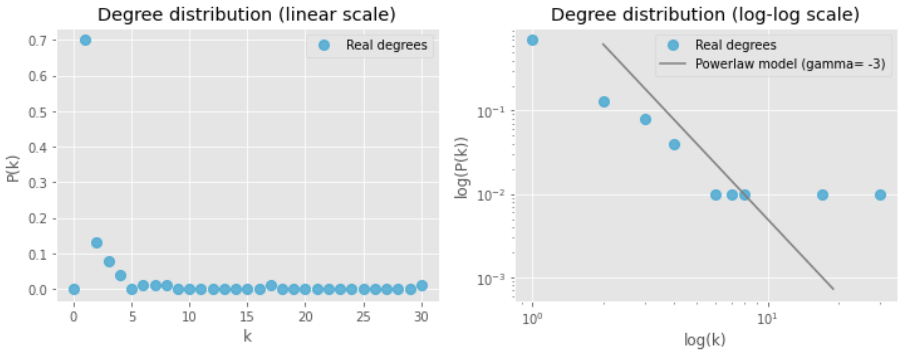

Figure 12: BA with parameters: n=100, n_0=4, m=1

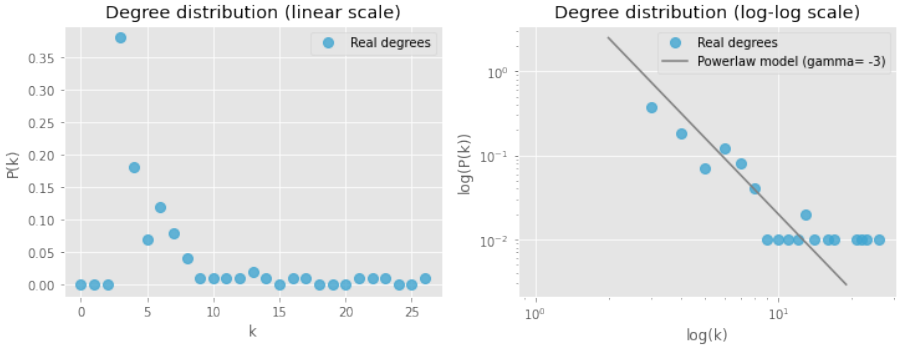

Figure 13: BA with parameters: n=100, n_0=4, m=2

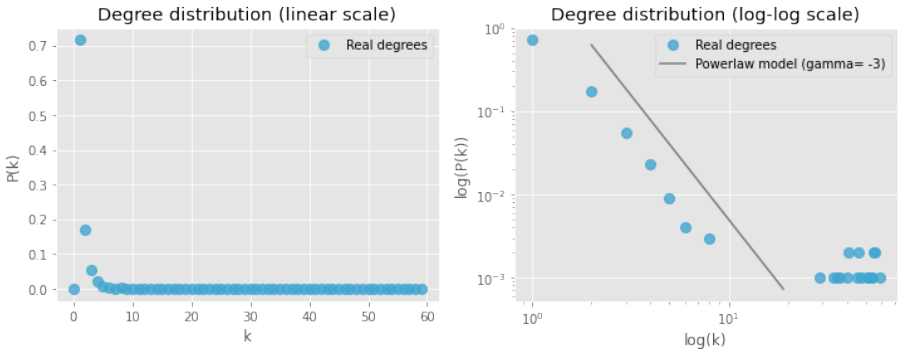

Figure 14: BA with parameters: n=1000, n_0=20, m=1

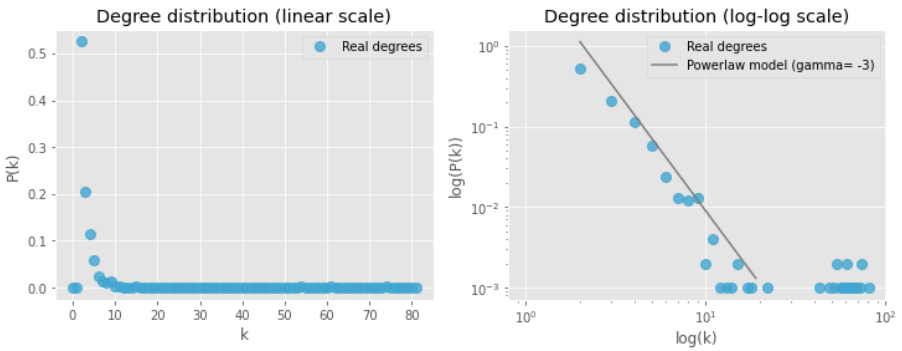

Figure 15: BA with parameters: n=1000, n_0=20, m=2

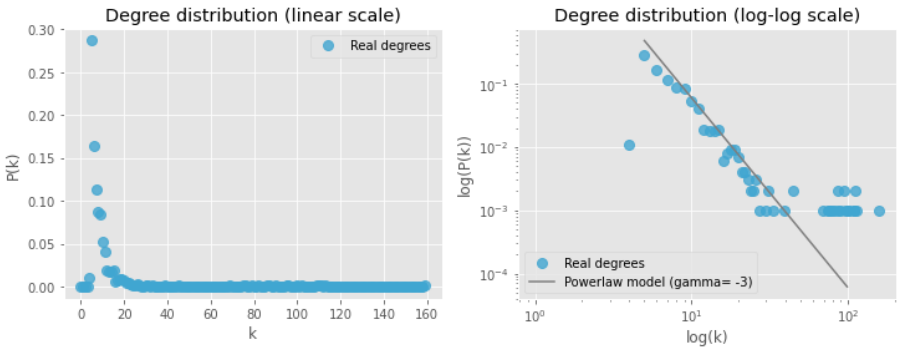

Figure 16: BA with parameters: n=1000, n_0=20, m=5

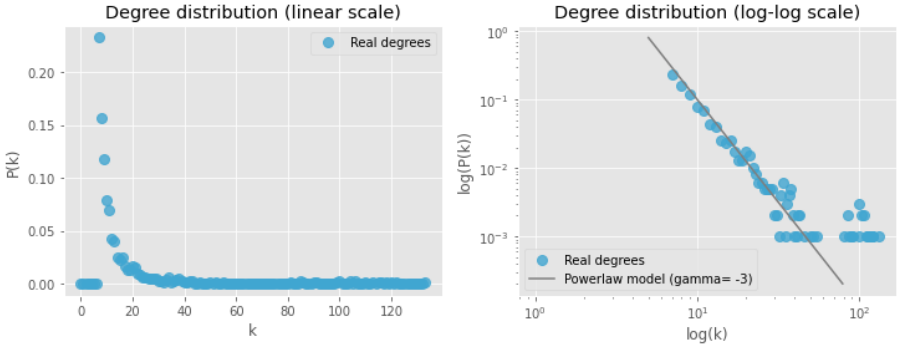

Figure 17: BA with parameters: n=1000, n_0=20, m=7

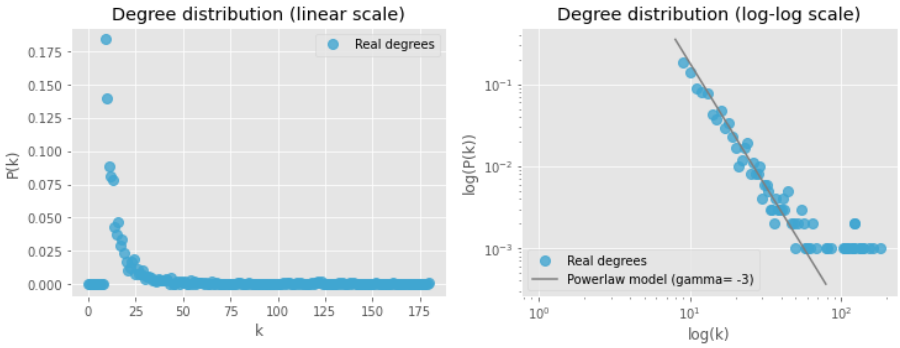

Figure 18: BA with parameters: n=1000, n_0=20, m=9

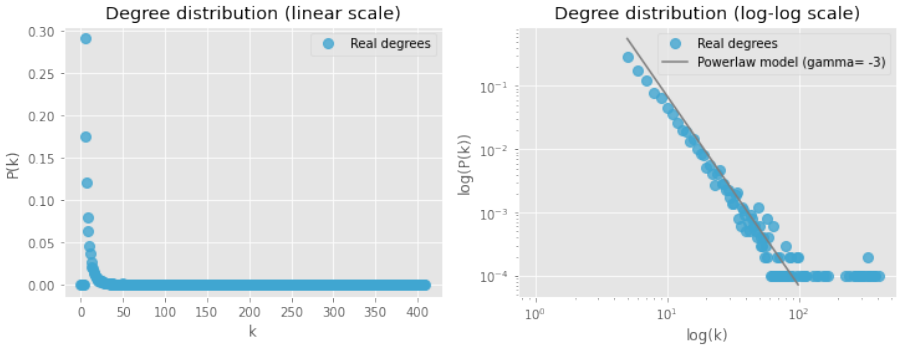

Figure 19: BA with parameters: n=10000, n_0=20, m=5

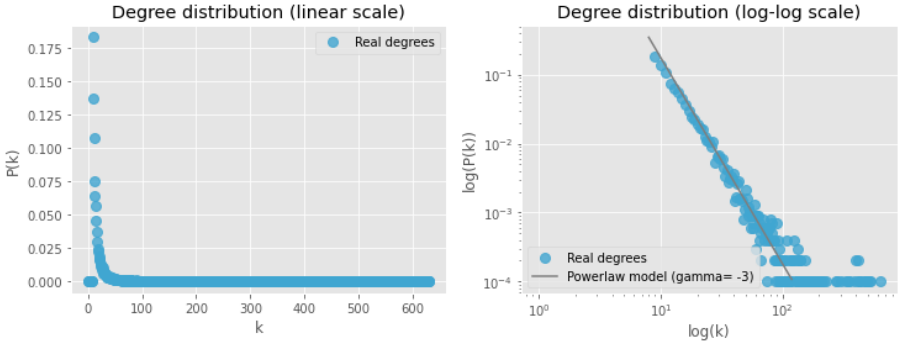

Figure 20: BA with parameters: n=10000, n_0=20, m=9

From the above plots we can see that by increasing the size of the network, the power-law distribution with degree -3 becomes more accentuated, while the effect of the m parameter is not easy to see on these plots.

4.4 Estimation of the exponent for the empirical degree distributions

In case of a scale-free network, i.e. a network following a power-law degree distribution (as the ones generated by a BA model), we may be interested in estimating the exponent of the empirical power-law distribution. We did this in the case of the generated scale-free networks with a number of nodes greater that 1000.

In particular, we used the 3 suggested approaches to perform the estimation:

- MLE
- Linear regression on the PDF histogram with log bins
- Linear regression on the CCDF histogram with log bins

All the details of the implementation of these 3 methods are accurately described in the final part of the notebook *BA_model.ipynb*.

For each scale-free model we built, with the BA algorithm we performed the estimation of the γ exponent by using all the 3 methods and we reported the results into the table 2

n_0	m	N	bins	PDF	CCDF	MLE
20	1	1000	15	2.118	1.987	2.427
20	2	1000	15	2.309	2.189	2.475
20	5	1000	15	2.523	2.67	2.775
20	7	1000	15	2.565	2.666	2.813
20	9	1000	15	2.774	2.833	2.823
20	5	10000	15	2.824	2.713	2.895
20	9	10000	15	2.837	2.864	2.897

Table 2: Estimation of the exponent for the empirical degree distributions of BA

The above table clearly shows that binning-based approaches (i.e. consisting on a linear regression based on the PDF or CCDF histograms, with logarithmic bins), perform poorly. As we can easily imagine, in binning-based approaches the estimated exponent is highly dependent on the choice of bin width (here we used a fixed number of 15 bins), and this dependency varies as a function of sample size as can be found in the "On estimating the exponent of power-law frequency distributions" paper [4].

In general, binning results in a loss of information about the distributions of points within a bin and is thus expected to perform poorly. Therefore, while binning is useful for visualizing the frequency distribution, binning-based approaches should be avoided for parameter estimation (see the paper "Power-law distributions in empirical data" [5]).

Maximum likelihood estimation performs the best in estimating the powerlaw exponent, and it is perfectly reflected in our table, where can see that MLE always estimates values closer to 3.

What we can see is also that the size of the network N and the parameter m improve the estimation for all the methods and particularly the CCDF based estimate overcomes the PDF ones by increasing m.

[4] White, Ethan P., Brian J. Enquist, and Jessica L. Green. "On estimating the exponent of power-law frequency distributions". Ecology 89.4 (2008): 905-912.

[5] "Power-Law Distributions in Empirical Data" Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman SIAM Review 2009 51:4, 661-703.

5 Configuration model (CM)

The CM algorithm generates random networks from a given degree sequence.

5.1 Generation of the network

The detailed steps of our implementation of the algorithm are showed in the notebook *CM_model.ipynb*.

We just briefly explain the main inputs to the CM algorithm:

- N: number of nodes
- p_k: nodes degree distribution

The output of the algorithm will be a complex network whose nodes follow the desired degree distribution.

In particular, in our implementation we used two types of distribution:

- Poisson degree distribution with parameter λ, which allows to generate an ER network
- Power-law degree distribution with parameter γ, which allows to generate a scale-free networks

By changing these parameters, i.e. the number of nodes and the chosen distribution parameter, the algorithm will build a network accordingly to the chosen values.

All the choices we made for what regards the generation of the node degrees for each node, the building of the vector of "stubs"[6], its random permutation, and the creation of links among the nodes are described in detail inside the provide notebook.

What we would like to remark is the way we managed the appearance of multiple edges and self-loops. In the case of a power-law degree distribution we inserted a simple check on the value of γ that verifies whether it is included between 2 and 3: if this happens we adopted a "cutoff" strategy in the generation of node degrees, by using as a maximum value for the degree of a node the square root of the total number of nodes of the desired final network. This prevention allowed to limit the number of self-loops and parallel edges and in the end, after executing the algorithm, we decided to remove those edges, since they were present in an irrelevant number with respect to the total number of edges and nodes.

5.2 Overview of the generated networks

During our experiments, we generated networks of different sizes but let's see first the plots of some of the small-size networks we generated.

5.2.1 Networks following a Poisson distribution

For what regards networks generated by using as input a Poisson degree sequence, we analyzed the influence of the λ parameter on the structure of the network.

The figure 21 demonstrates that, by fixing the size of the network to 50 nodes, the higher γ is, the more dense and also connected the network is, indeed the number of single-node components reduces a lot. Those 2 networks can be found under the names
CM_poisson_lambda2#0_n50.net, *CM_poisson_lambda4#0_n50.net*, under the *nets* folder.

In figure 22 we report 2 similar networks, but with a size of 100 nodes. We can easily draw the same conclusions as above.

[6] half-edges of the nodes, later used for binding two nodes and thus forming a proper edge

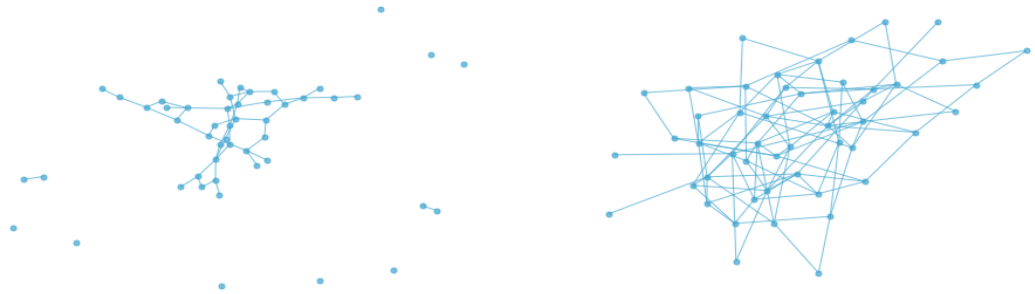

Figure 21: CM networks with 50 nodes following a Poisson degree distribution with λ set to 2 on the left and to 4 on the right

Figure 22: CM networks with 100 nodes following a Poisson degree distribution with λ set to 2 on the left and to 4 on the right

Those 2 networks can be found under the names *CM_poisson_lambda2#0_n100.net*, *CM_poisson_lambda4#0_n100.net*, under the *nets* folder.

5.2.2 Networks following a Power-law distribution

For what regards networks generated by using as input a Power-law degree sequence, we analyzed the influence of the γ parameter on the structure of the network.

The figure 23 demonstrates that, by fixing the size of the network to 100 nodes, the lower γ is, the more dense and also connected the network is, thus reducing the number of single-node components. Those 3 networks can be found under the names *CM_powerlaw_gamma2#2_n100.net*, *CM_powerlaw_gamma2#6_n100.net*, *CM_powerlaw_gamma3#5_n100.net*, under the *nets* folder.

Figure 23: CM networks with 100 nodes following a Power-law degree distribution with gamma set to 2.2 on the left, 2.6 on the right, and 3.5 on the bottom net

5.3 Degree distributions and modelisation

In this part, we will make observations about the experimental degree probability distribution and confirm the theoretical modelisation.

5.3.1 Networks following a Poisson distributions

We know that the degree distribution of a CM networks reflects the degree sequence given as input to the algorithm. For what regards models generated by using a Poisson degree probability distribution,we should observe a Poisson curve in the plots.

Let's visualize some of these distributions in the figures 24, 25, 26, 27, 28, 29, 30, 31, 32, 33.

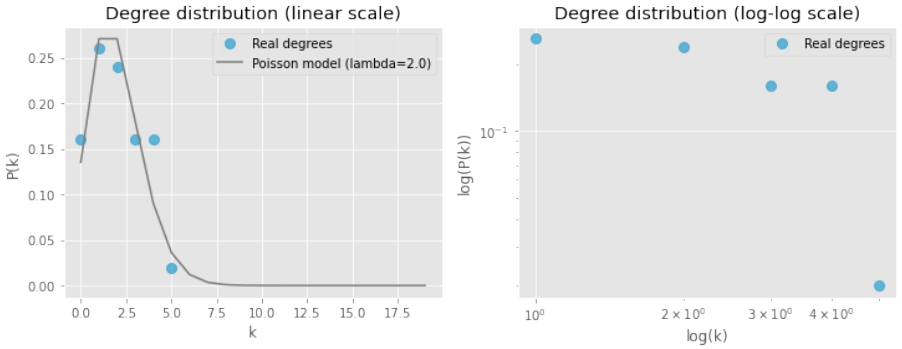

Figure 24: CM based on Poisson distribution with parameters: n=50, λ=2

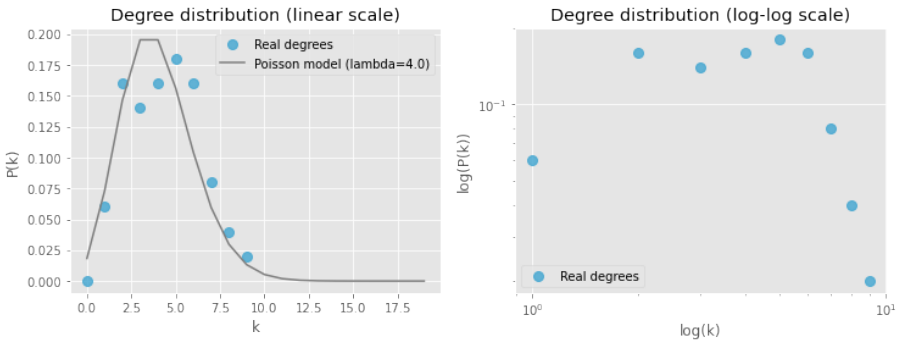

Figure 25: CM based on Poisson distribution with parameters: n=50, λ=4

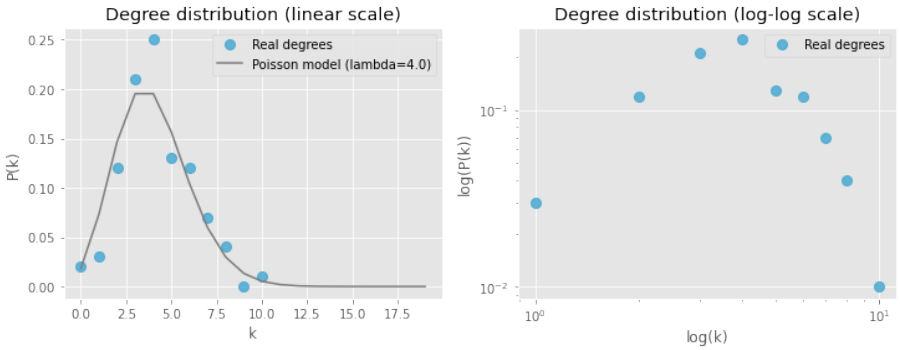

Figure 26: CM based on Poisson distribution with parameters: n=100, λ=4

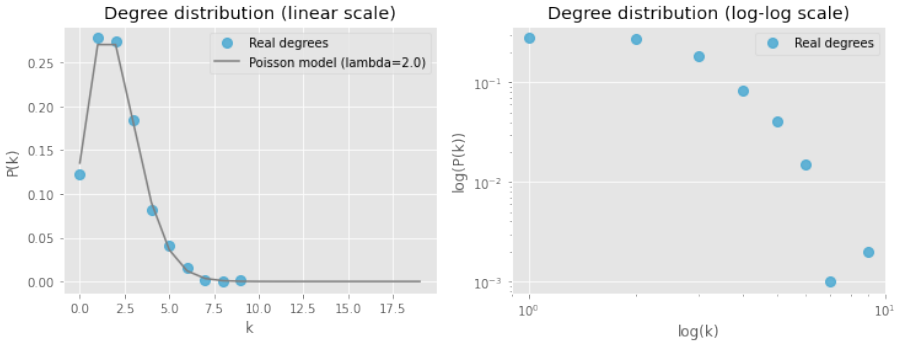

Figure 27: CM based on Poisson distribution with parameters: n=1000, λ=2

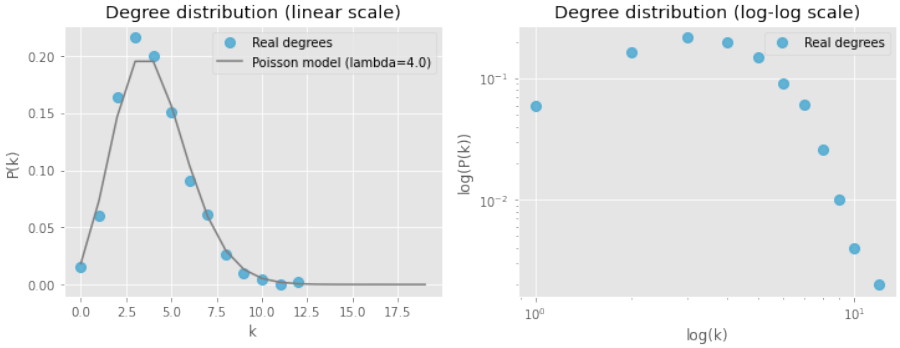

Figure 28: CM based on Poisson distribution with parameters: n=1000, λ=4

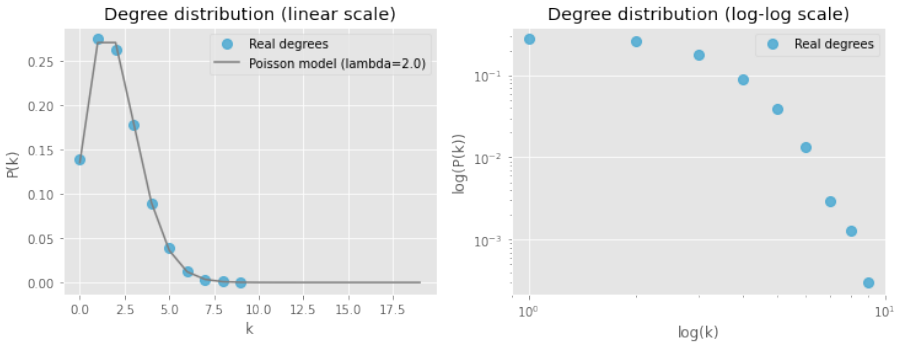

Figure 29: CM based on Poisson distribution with parameters: n=10000, λ=2

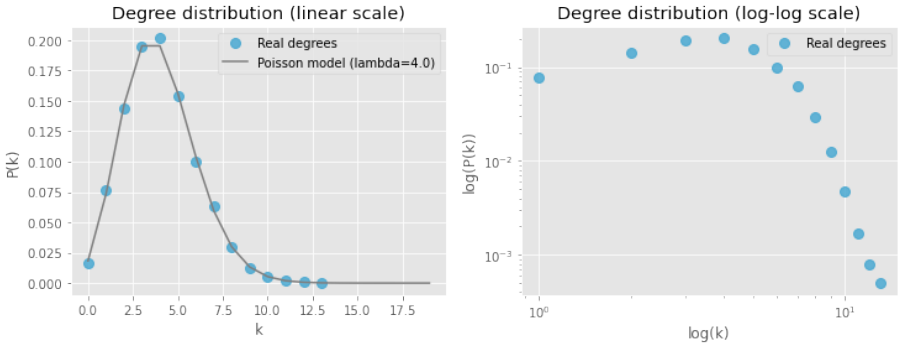

Figure 30: CM based on Poisson distribution with parameters: n=10000, λ=4

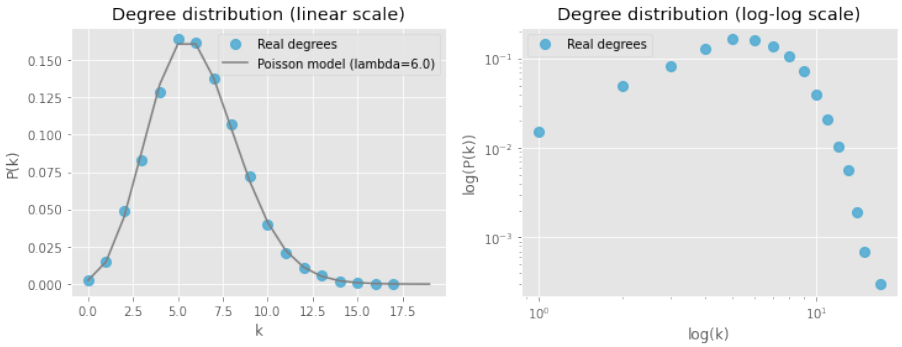

Figure 31: CM based on Poisson distribution with parameters: n=10000, λ=6

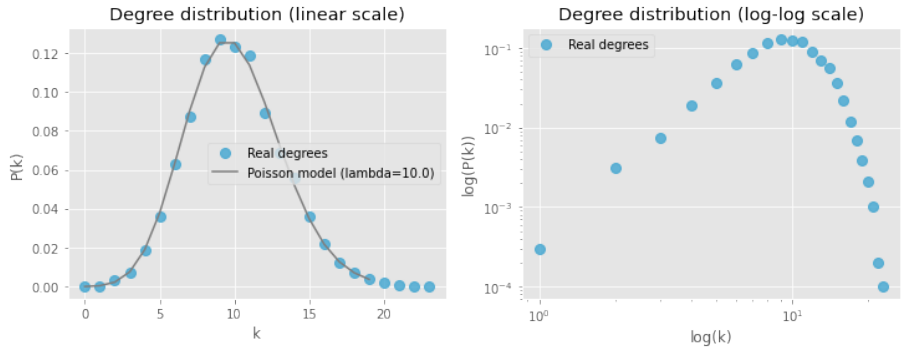

Figure 32: CM based on Poisson distribution with parameters: n=10000, λ=10

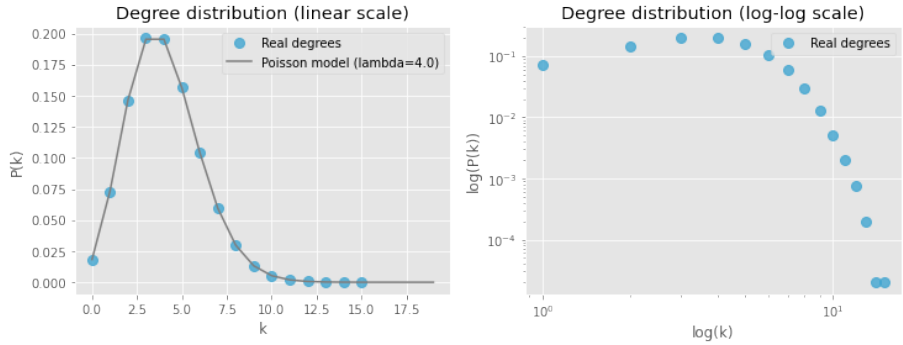

Figure 33: CM based on Poisson distribution with parameters: n=50000, λ=4

From the above plots we can see that by increasing the size of the network, Poisson distribution becomes more accentuated, while the effect of the λ parameter simply changes the mean of the Poisson curve. We can also see that the experimental distribution fits quite well the theoretical one when the size N of the network increases.

5.3.2 Networks following a Power-law distributions

For what regards CM models generated by using a power-law degree probability distribution, we should observe a scale-free curve in the distribution plots, i.e. a linear curve in log-log scale.

Let's visualize some of these distributions in the figures 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47.

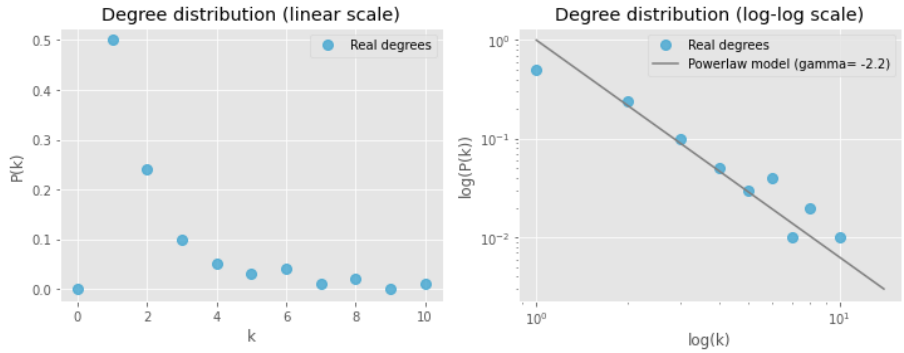

Figure 34: CM based on power-law distribution with parameters: n=100, gamma=2.2

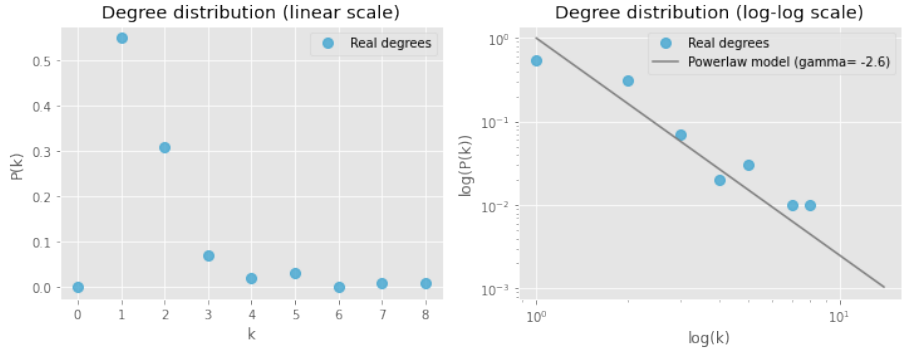

Figure 35: CM based on power-law distribution with parameters: n=100, gamma=2.6

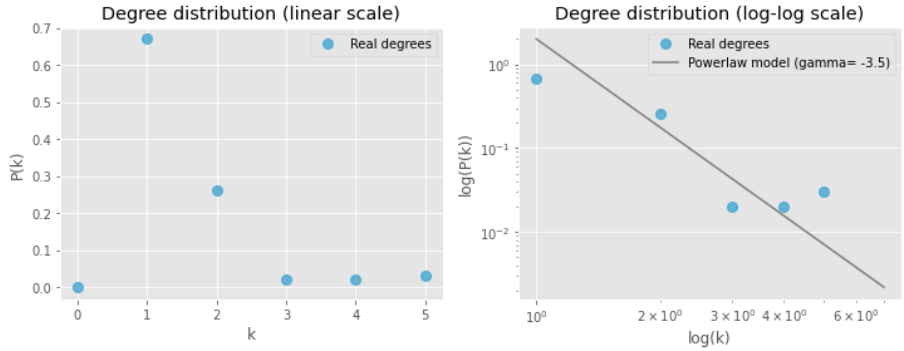

Figure 36: CM based on power-law distribution with parameters: n=100, gamma=3.5

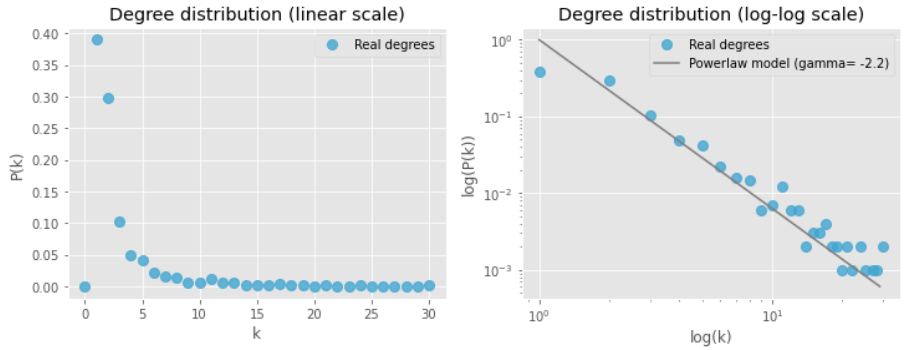

Figure 37: CM based on power-law distribution with parameters: n=1000, gamma=2.2

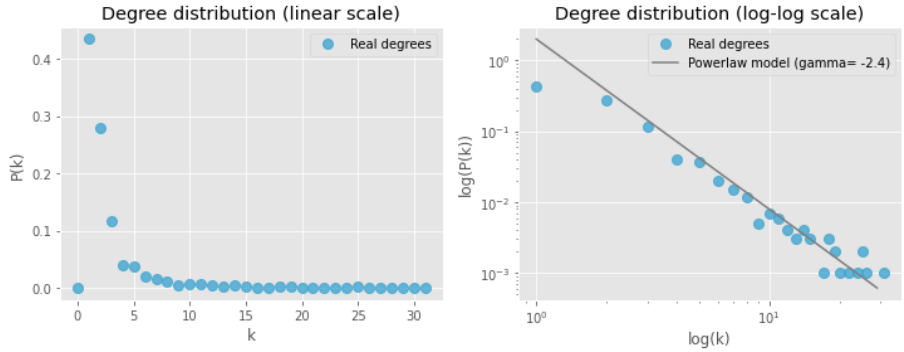

Figure 38: CM based on power-law distribution with parameters: n=1000, gamma=2.4

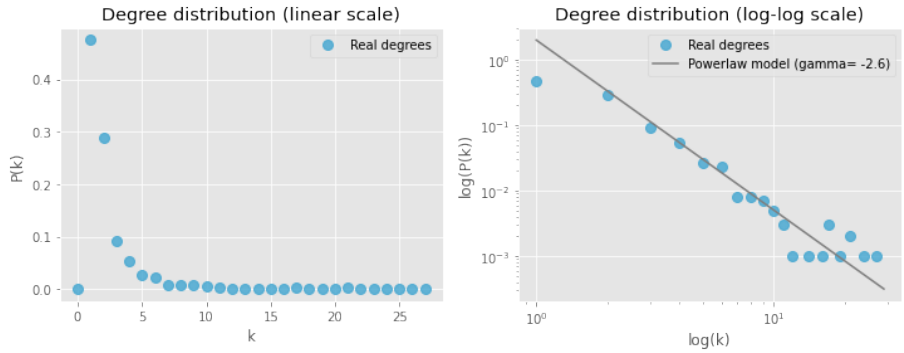

Figure 39: CM based on power-law distribution with parameters: n=1000, gamma=2.6

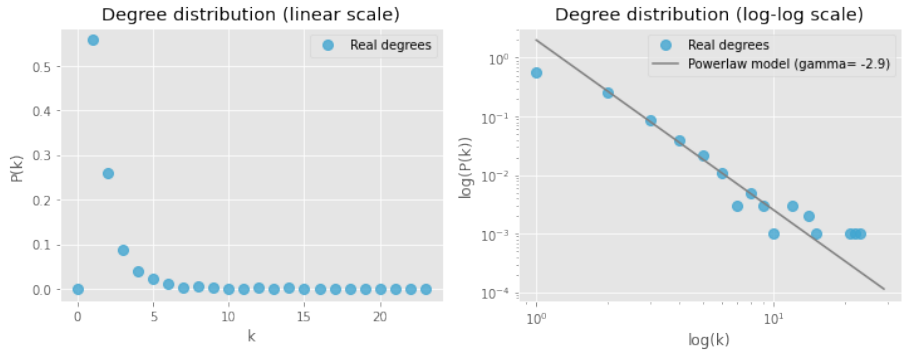

Figure 40: CM based on power-law distribution with parameters: n=1000, gamma=2.9

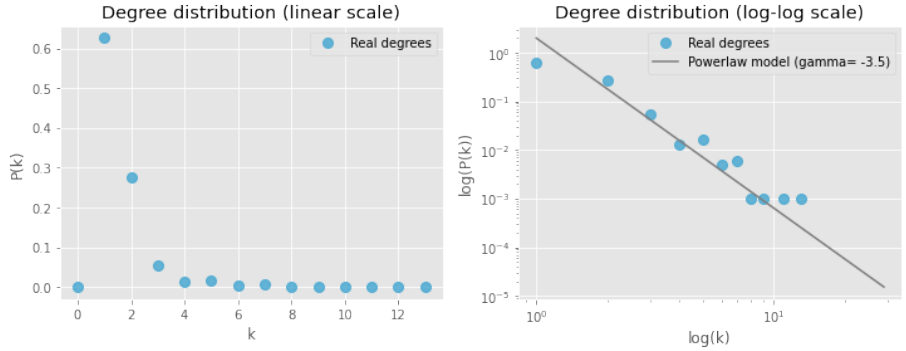

Figure 41: CM based on power-law distribution with parameters: n=1000, gamma=3.5

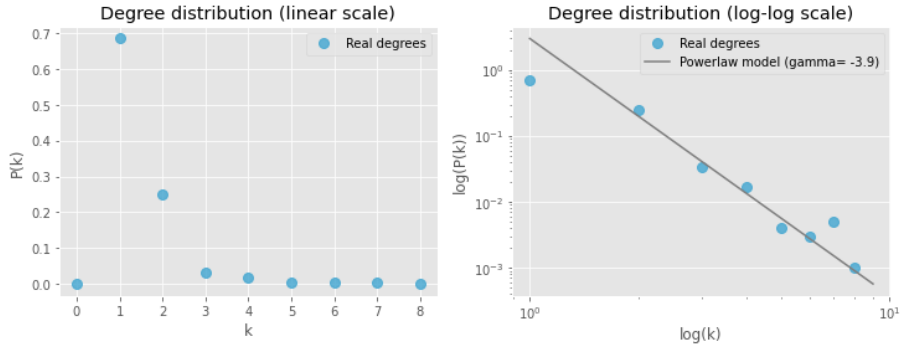

Figure 42: CM based on power-law distribution with parameters: n=1000, gamma=3.9

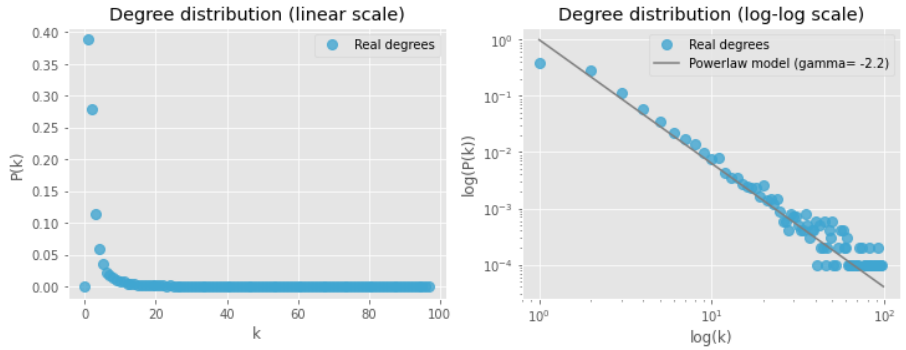

Figure 43: CM based on power-law distribution with parameters: n=10000, gamma=2.2

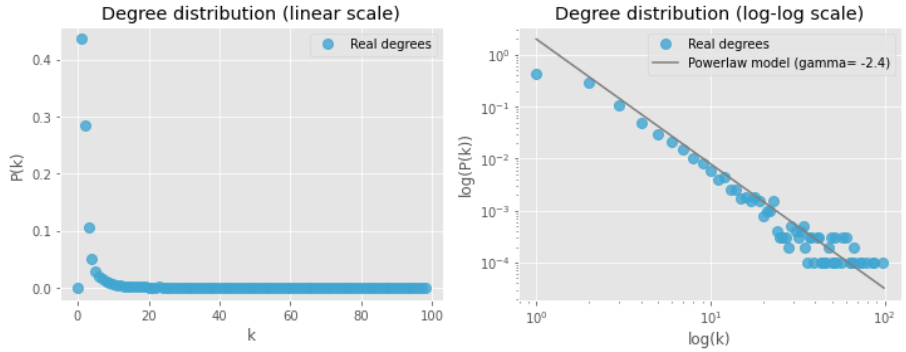

Figure 44: CM based on power-law distribution with parameters: n=10000, gamma=2.4

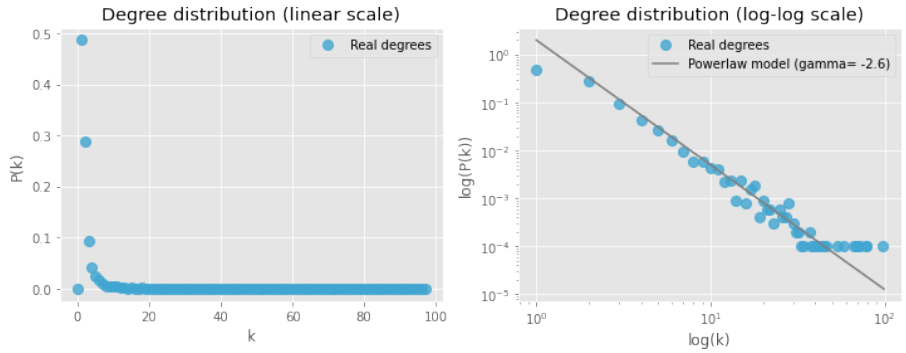

Figure 45: CM based on power-law distribution with parameters: n=10000, gamma=2.6

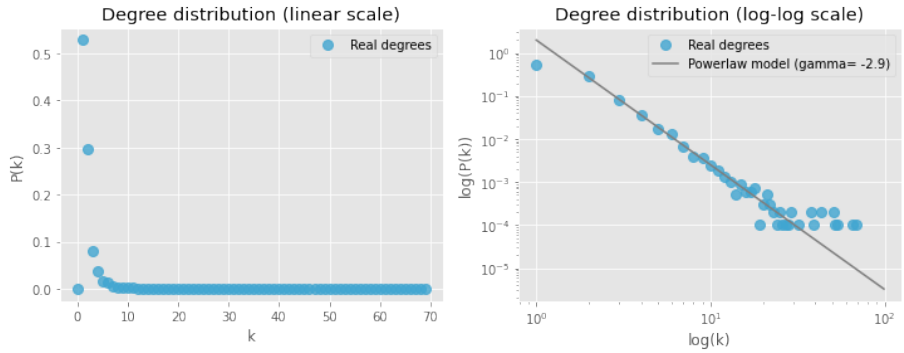

Figure 46: CM based on power-law distribution with parameters: n=10000, gamma=2.9

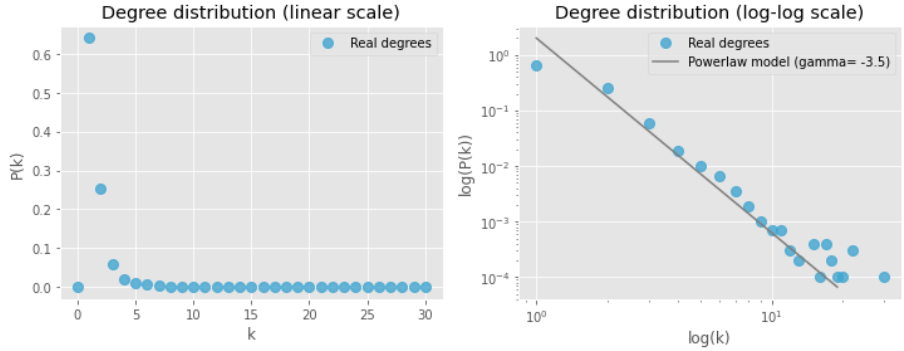

Figure 47: CM based on power-law distribution with parameters: n=10000, gamma=3.5

From the above plots we can see that by increasing the size of the network, the power-law distribution becomes more accentuated, while the effect of the gamma is not easy to see.. We can also see that the experimental distribution fits quite well the theoretical one when the size N of the network increases.

5.4 Estimation of the exponent for the empirical degree distributions

We did the same reasoning done for BA models for what regards the estimation of the exponents of scale-free networks, i.e. those generated by using as a parameter a power-law degree distribution with a given γ. We therefore performed the estimation of the exponents according to the same 3 techniques seen for the BA model. Again, all the details of the implementation of these 3 methods are accurately described in the final part of the notebook *CM_model.ipynb*.

For each scale-free model with more than 1000 nodes that we built, we performed the estimation of the γ exponent and we reported the results into the table 3. The table reports the exponent estimations for different values of the parameter N, i.e. the size of the network, and also for different values of γ.

N	gamma	bins	PDF	CCDF	MLE
1000	2.2	15	2.424	2.441	2.304
1000	2.4	15	2.786	2.866	2.609
1000	2.6	15	2.638	2.658	2.493
1000	2.9	15	3.072	3.115	2.854
1000	3.5	15	3.7	3.494	3.602
1000	3.9	15	3.888	3.883	3.883
10000	2.2	15	2.33	2.466	2.274
10000	2.4	15	2.466	2.565	2.411
10000	2.6	15	2.71	2.806	2.555
10000	2.9	15	2.944	2.953	2.987
10000	3.5	15	3.446	3.31	3.36

Table 3: Estimation of the exponent for the empirical degree distributions of CM (SF)

5.4.1 Interpretation

We can make similar considerations with those we did in the case of the BA algorithm. Again, MLE, on average, gives better estimations for the power-law exponent.

What we can see, in this case, is that for high N, such as 10.000, or also for γ above 2.4 or 2.6, CCDF-based and PDF-based estimations are comparable and even better with respect to the MLE ones.