# Detection of Small-World Networks

Using the Spectral Test and the Tracy-Widom Distribution

E.K. Smit

**Bachelor thesis**
2022-2025

TU Delft

# Detection of Small-World Networks

## Using the Spectral Test and the Tracy-Widom Distribution

by

E.K. Smit

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Wednesday December 17th, 2025 at 11:00 AM.


| | | |
|---|---|---|
| Student number: | 5734762 | |
| Project duration: | August 2025, – December 2025 | |
| Thesis committee: | dr. N. Parolya, | TU Delft, supervisor |
| | Prof. Dr. Y. Blanter, | TU Delft, supervisor |
| | Dr. S. Eijt, | TU Delft, committee |
| | Dr. J. Dubbeldam | TU Delft, committee |


An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

# Preface

This thesis is the final part of my double Bachelor's degree in Applied Physics and Applied Mathematics at Delft University of Technology. During my studies, statistics was always one of my favorite subjects, so when I had the opportunity to do statistical research, I seized it. Through this thesis, I learned that actual research does not come with an answer sheet and that you often need to develop new solutions to the problems you encounter. I hope this thesis is the cherry on top of an, at times, difficult but very rewarding Bachelor's degree.

Finally, I would like to thank my supervisors, Dr. N. Parolya and Prof. Dr. Y. Blanter, who supported me and guided me through my research, as well as my friends and family, whose encouragement has helped me reach this point.

*E.K. Smit*
*Delft, December 2025*

# Abstract

Since graphs represent many real-world networks, understanding their mathematical properties is essential to analyze, modify, and predict their behavior. To characterize these properties, one must first identify the type of graph under study — a task that is not always straightforward. This paper examines one such model, the Watts–Strogatz small-world network, and investigates how well it can be distinguished from the classical Erdős–Rényi random graph. This is because a Watts–Strogatz network can be seen as an interpolation between a completely structured graph and a completely random graph. Two statistical tests are considered: the Spectral Test, as presented by Cai et al. (2017), and a new test based on the Tracy–Widom distribution. These tests are applied to controlled data with known parameters, allowing their accuracy to be quantitatively evaluated. Both tests exhibit comparable power; however, only the Tracy–Widom Test maintains a controlled significance level $\alpha$, making it the more reliable and preferable choice.

# Contents

# 1 Introduction

## 1.1. Networks

"You are only six connections away from every other person on the planet." Most people have heard this saying, yet few know that it is based on mathematical graph theory. The concept was first tested by Stanley Milgram in *An Experimental Study of the Small World Problem*, where participants were asked to send a package to a stranger using only their social network. On average, the number of steps needed was 5.2 Travers and Milgram (1969). This phenomenon is explained by Watts and Strogatz (1998), by the coexistence of high clustering and a few long-range ties (hubs). Big cities, for example, act as hubs with many connections, making it possible to link even the most separated people. Later, it was discovered that adding just a small number of random shortcuts to a highly clustered structure greatly reduces the number of steps needed to connect individuals.

Many systems in daily life can be represented as networks of components and their connections. Examples include friendship structures (with friends being the components and how well they know each other the connections), power grids (where the units are locations and the connections are routes between them), airport routes (airports being the components and the flight routes the paths), and the spread of infectious diseases (similar to the friendship system: people being the components and the connection is who infected whom). To understand how such a network behaves, it is essential to study its underlying mathematical characteristics. Consider, for example, the spread of a virus in a population (examples used by Newman (2002); Pastor-Satorras and Vespignani (2001)): predicting how quickly it spreads, how long quarantines should last or how much vaccination is needed requires knowledge of the structural features of the system. In epidemiology, the effectiveness of vaccination strategies and the epidemic threshold depend on whether the underlying contact network has small-world or scale-free properties—concepts that will be discussed later in this thesis.

However, identifying the traits of a given network, especially when that network is large, is not always straightforward. Some systems, such as *small-world networks*, are difficult to distinguish conclusively. This is because these structures have features from several other types of networks, making it difficult to determine the appropriate network model. Although there are several tests, none are definitive enough to reliably determine the properties of a network. Developing such tests is important, since many real-world networks likely exhibit small-world characteristics. Examples include human social systems, neural connections in the nematode C. elegans, the Western US power grid, and networks of scientific collaborations (as presented by Amaral et al. (2000)). This paper focuses on identifying small-world networks by comparing different test statistics to determine which provide the most accurate results. First, important definitions are introduced, followed by an explanation of the tests used, and finally, the results are presented and discussed.

Graphs provide the standard mathematical framework for describing complex systems, which is necessary to test them. They are often used to visualize datasets that are too large or complex to analyze with traditional methods (Slutsky (2014)). In this framework, each component of the system is represented by a node, and each connection between components is represented by an edge. Graphs come in many forms, but this thesis focuses on three types: small-world graphs, Erdős–Rényi random graphs, and regular ring lattices. This is because small-world structures have characteristics of both

random graphs and regular ring lattices. Two key properties of a network are the average path length $L$, which is the average number of steps required to get from one node to another, and the clustering coefficient $C$, which measures how strongly the structure is clustered. A cluster is a group of nodes with more links between themselves than with the rest of the system. Formally, as stated by Tsourakakis (2008), the clustering coefficient of a node is defined as the ratio of the number of existing edges among its neighbors to the total number of possible links. The clustering coefficient of the graph is then the average of these node-level coefficients. A graph with a visible cluster structure can be seen in Figure 1.1.

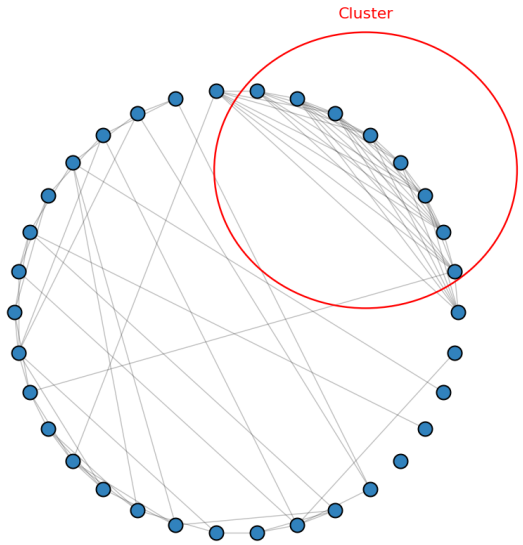


Figure 1.1: A network with a cluster structure within the red circle.

### 1.1.1. Regular Ring Lattice graph (RRL)

A *regular ring lattice* (RRL) is a circular graph in which each node is connected to its $k$ nearest neighbors. More formally: an RRL $C_n^k$, where $n \geq 4,\ 1 \leq k < \lfloor n/2 \rfloor$, is an undirected graph (connections go both ways) with $n$ vertices and a circulant adjacency matrix $\mathbf{A}$ generated by a vector $\mathbf{w} \in \{0,1\}^n$, with components (see Fabris (2023))

$$w_l = \begin{cases} 1, & \text{if } l \in \{2, \dots, k+1\} \cup \{n-k+1, \dots, n\}, \\ 0, & \text{otherwise.} \end{cases}$$

In such graphs, the average path length $L$ grows linearly with $n$. Because each node is only connected to its nearest neighbors, reaching distant nodes requires traversing a number of edges proportional to the network size. Thus, larger networks have proportionally longer typical paths. At the same time, because neighbors are strongly interconnected, the network exhibits significant clustering, given by $C = \frac{3(k-2)}{4(k-1)}$ from Meghanathan (2015). An example is shown in Figure 1.2. With adjacency matrix:

$$A_{n=10,k=4} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

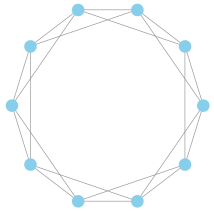

Figure 1.2: An RRL with $n = 10$ and $k = 4$.

It is a classical result that every $n \times n$ circulant matrix $C$, which is a matrix in which every row is obtained by shifting the previous row one position to the right, has eigenvalues

$$\lambda_r(C) = \sum_{m=0}^{n-1} c_m \, \omega_n^{rm}, \qquad r = 0, 1, \dots, n-1, \quad \omega_n = e^{-2\pi i/n}. \tag{1.1}$$

This formula is explicitly stated in Chauhan et al. (2009). The adjacency matrix $A_{\mathrm{RRL}}$ of a regular ring lattice is circulant, meaning that all the rows are shifted versions of the first row by one entry. The ones are in the $\pm 1, \pm 2, \dots, \pm \frac{k}{2}$ positions of the first row. Substituting these entries into (1.1) yields the well-known cosine form:

$$\lambda_r(A_{\mathrm{RRL}}) = 2 \sum_{s=1}^{k/2} \cos\left(\frac{2\pi r s}{n}\right), \qquad r = 0, 1, \dots, n-1.$$

Hence, the spectrum is explicitly determined. This spectrum will be necessary for the statistical tests described in later sections.

### 1.1.2. Erdós-Rényi (ER)

An *Erdős–Rényi (ER)* graph is constructed by connecting each pair of nodes independently with probability $p$. Each entry $(i, j)$ in its adjacency matrix follows a Bernoulli distribution with parameter $p$. These graphs have an average path length that increases logarithmically with $n$, indicating that even in large networks, it remains relatively easy to reach any node from another. However, because the graph lacks an inherent structure, it exhibits very little clustering: no group of nodes is significantly more interconnected than the rest, and the clustering coefficient is $C = \frac{k}{n}$, as found in Meghanathan (2015). An example of such a graph is shown in Figure 1.3, along with its corresponding adjacency matrix.

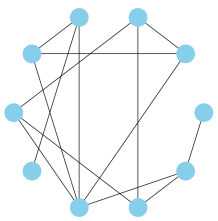

Figure 1.3: An ER graph with $n = 10$ and $p = 0.3$.

$$A_{n=10,p=0.3} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

This is a symmetric adjacency matrix with no self-loops (a self-loop is when a node connects to itself), so all diagonal entries are zero. The eigenvalues of the adjacency matrix of an ER graph obey Wigner's semicircle law: all but the largest eigenvalue lie within the interval

$$[-2\sqrt{np(1-p)},\ 2\sqrt{np(1-p)}],$$

known as the *bulk*. The largest eigenvalue of this matrix, $\lambda_{\max}$, lies outside this bulk and is of order $np$, given by Erdős et al. (2013). This difference between the bulk and $\lambda_{\max}$ comes from the fact that an ER graph can be separated into two components: a regular, predictable part and a random part. The predictable part represents the average level of connectivity across the network, while the random part captures the small, irregular variations in how individual nodes are connected. When combined, these two effects produce a spectrum where most eigenvalues cluster together in a broad "bulk," while a single large eigenvalue appears separately. The bulk reflects the random fluctuations in the graph's structure, and the large eigenvalue corresponds to the overall average degree of connectivity, which is approximately $np$.

### 1.1.3. Watts-Strogatz small-world graph (WS)

Small-world graphs are networks that combine short average path lengths with a high clustering coefficient. The concept of a small-world network originates from the experiment performed by Stanley Milgram. In this study, participants were asked to send a package to an unknown receiver, thousands of miles away, using only their own social contacts. Milgram found that most people needed six links or fewer to complete the task (Stanley Milgram (2025)). Several types of graphs exhibit small-world properties: scale-free networks, broad-scale networks, and Watts–Strogatz networks (among others presented by Amaral et al. (2000)).

A *scale-free network* is a graph in which the degree distribution follows a power law: $\mathbb{P}(k) \sim k^{-\gamma}$, showing that while the average degree of the graph is $k$, it varies highly per node. When constructing such a graph, new links are added preferentially to nodes that already have higher degree:

$$\mathbb{P}(\text{linking to node } i) \sim \frac{k_i}{\sum_j k_j}.$$

An example found in Weisstein (2025) is the network of Hollywood actors (nodes), with edges representing movies in which two actors have appeared together.

*Broad-scale networks* share the same general degree distribution as scale-free networks, but with a sharper cutoff (Amaral et al. (2000)).

Most relevant for this thesis are *Watts–Strogatz (WS)* graphs. They are generated by starting with a regular ring lattice (RRL) of $n$ nodes, each connected to its $k$ nearest neighbors, and then *rewiring* edges. First, each existing edge is removed with probability $\beta$. Next, each missing edge is added back with probability $\beta\frac{k}{n-1}$, an algorithm presented by Cai et al. (2017). See Figure 1.4. This type of graph differs from scale-free networks in two important ways. First, the edges in a WS graph are attached to nodes randomly, whereas the scale-free networks use preferential attachment to nodes with higher degree. Second, nodes in a WS graph all have degree close to the average degree $k$, unlike scale-free networks. See Marcozzi (2017). Normally, before the rewiring process, the ring lattice undergoes a permutation $\mathcal{P}_\pi \in \{0,1\}^{n\times n}$ so that the adjacency matrix becomes $A = \mathcal{P}_\pi B \mathcal{P}_\pi^T$, with $B$ the regular ring lattice (again, Cai et al. (2017)). This operation rearranges the order of the nodes but does not

change the actual connections between them. The purpose of this step is to remove any dependence on the original node numbering, ensuring that the model treats all nodes equally before randomness is introduced through the rewiring process with probability $\beta$. Due to the computational difficulty of accounting for every permutation, this thesis assumes no permutation is performed prior to rewiring unless explicitly stated otherwise. The parameter $\beta$ can take any value between 0 and 1. When $\beta = 0$,

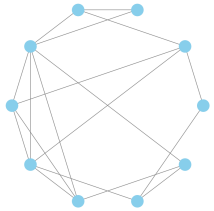

Figure 1.4: A WS graph, generated from the RRL in Figure 1.2, with $n = 10$, $k = 4$, and $\beta = 0.5$.

the graph is a regular lattice, while $\beta = 1$ corresponds to a purely random Erdős–Rényi (ER) graph, where $p = \frac{k}{n-1}$. This shows that WS graphs interpolate between the two extremes. They thus combine the properties of both: short average path lengths (as in ER graphs) and high clustering (as in RRLs). The clustering coefficient depends on $\beta$ (see Meghanathan (2015)):

$$C(\beta) = \frac{3(k-2)}{4(k-1)}(1-\beta)^3 \;=\; C(0)(1-\beta)^3.$$

Applications of WS graphs include social networks (where individuals have close circles of friends yet can still connect widely), global airline routes, and neural networks. However, when looking at an adjacency matrix, it is hard to tell if the graph is completely random or has small-world structure.

This thesis investigates two statistical tests to determine how accurately a WS graph can be distinguished from an ER graph. In both tests, $H_0$ is that the network is ER ($\beta = 1$), while $H_1$ means the graph has small-world properties ($\beta < 1$).

We approach this question using two detection methods, first applied to artificial data where $n$, $k$, and $\beta$ are known. This allows us to study the size and power of each test. Both tests are framed in terms of hypothesis testing: under the null hypothesis the graph follows an ER model ($\beta = 1$), while under the alternative the model is WS ($\beta < 1$). The WS model, presented by Watts and Strogatz (1998) requires $\ln n \ll k \ll n$ to ensure that the rewired graph remains connected. The condition serves to balance two competing effects in the model. On one hand, if $k$ is too small, many nodes will have too few links to stay connected after the rewiring step, causing the network to break apart into disconnected pieces. On the other hand, if $k$ is too large, the graph becomes almost fully connected, losing the sparse, local structure that characterizes small-world networks. The double inequality $\log(n) \ll k \ll n$ ensures that the network is dense enough to stay connected, yet sparse enough to retain meaningful local clustering and short path lengths.

## 1.2. Detection of Networks

### 1.2.1. Detection regions

According to Cai et al. (2017), the parameter $\beta$ can fall into three regimes, each determining the detectability of small-world structure: the *impossible region*, the *hard region*, and the *easy region*.

When

$$1 - \beta \lesssim \left( \sqrt{\frac{\log n}{n}} \vee \frac{\log n}{k} \right),$$

where $\lesssim$ stands for asymptotically smaller than and $\vee$ takes the maximum of two options. The constant $\beta$ belongs to the impossible region and is asymptotically too close to an ER network for any test to detect the difference and confidently reject $H_0$. This means that when $\beta$ is very close to one, almost all edges have been rewired, and the original lattice structure has been lost. The graph then behaves

just like a random Erdős–Rényi network with the same connection probability. Because both models generate nearly identical structures, no statistical method—no matter how sophisticated—can distinguish between them as the network grows large. In this sense, the distinction between "small-world" and "random" becomes meaningless in the impossible region and the test error never converges to zero. The $\lesssim$ sign means that $1-\beta$ is asymptotically smaller than the right-hand side of the equation. Formally, we have the following theorem.

**Impossible region:** Consider the following statistical models: $\mathbb{P}_0$ denotes the distribution of an Erdős–Rényi random graph with $n$ nodes and $p = \frac{k}{n-1}$ (the region where $H_0$ is true), and $\mathbb{P}_\pi$, $\pi \in S_{n-1}$ ($S_{n-1}$ is the set of possible permutations) denote distributions of a Watts–Strogatz small-world graph with $n$ nodes, connected to its $k$ nearest neighbours and rewiring probability $\beta$, with different permutations $\pi$ (the region where $H_1$ is true).

Consider any selector

$$\phi : \{0,1\}^{n\times n} \to S_{n-1} \cup \{0\}$$

that maps the adjacency matrix $A \in \{0,1\}^{n\times n}$ to a permutation in $S_{n-1}$ or to $\{0\}$ when no permutation is found. When $\phi = 0$ the test does not reject $H_0$ and when $\phi = \pi$, it matches the network to one of the WS permutations and rejects $H_0$. Then for any fixed $0 < \epsilon < \frac{1}{8}$, the following lower bound on multiple testing error holds:

$$\lim_{n\to\infty} \min_{\phi} \left\{ \mathbb{P}_0(\phi \neq 0) \vee \frac{1}{(n-1)!} \sum_{\pi\in S_{n-1}} \mathbb{P}_\pi(\phi \neq \pi) \right\} \geq 1 - 2\epsilon.$$

Whenever

$$1-\beta \lesssim \left( \sqrt{\frac{\log n}{n}} \vee \frac{\log n}{k} \right).$$

The term $\mathbb{P}_0(\phi \neq 0)$ means that $H_0$ is true, but the test rejects the null, while $\frac{1}{(n-1)!}\sum_{\pi\in S_{n-1}} \mathbb{P}_\pi(\phi \neq \pi)$ does the opposite for every permutation. This line states that even for $n \to \infty$, the best selector $\phi$ cannot control the error when this bound applies. Understanding what this region implies for certain $n$ and $k$ is crucial because every test here will fail. It is therefore impossible to design a test that succeeds in this regime. The relevant question becomes which test performs better in the other regions.

### 1.2.2. Hard interval and Maximum Likelihood

In the *hard region*, detection becomes possible whenever

$$1-\beta \gtrsim \left( \sqrt{\frac{\log n}{n}} \vee \frac{\log n}{k} \right),$$

where the testing error vanishes asymptotically.

**Detection: Easy and Hard Boundaries** Consider the same models as above. Minimax detection of the WS model is possible when

$$\lim_{n\to\infty} \min_{\phi} \left\{ \mathbb{P}_0(\phi \neq 0) \vee \frac{1}{(n-1)!} \sum_{\pi\in S_{n-1}} \mathbb{P}_\pi(\phi \neq 1) \right\} = 0.$$

This can be achieved using a maximum likelihood test, which searches for the permutation of the lattice structure that maximizes overlap with the observed adjacency matrix. If the graph contains an underlying lattice structure, this overlap will be significantly larger than in the purely random case. The corresponding test statistic $T_1(A)$ is defined as

$$T_1(A) := \max_{P_\pi} \langle P_\pi B P_\pi^T, A \rangle,$$

where $P_\pi \in \{0,1\}^{n\times n}$ ranges over all permutation matrices and $A$ is the observed adjacency matrix. The maximum likelihood test $\phi_1 : A \to \{0,1\}$ based on $T_1$ is given by

$$\phi_1(A) = \begin{cases} 1, & \text{if } T_1(A) \geq \frac{k}{n-1}nk + 2\sqrt{\frac{k}{n-1}nk \cdot \log n!} + \frac{2}{3}\log n!, \\ 0, & \text{otherwise.} \end{cases}$$

If $\phi = 1$, the graph is classified as a generated small-world graph and $H_0(\beta = 1)$ can be rejected. If $\phi = 0$, the null hypothesis cannot be rejected (Cai et al. (2017)). However, computing this test requires exponential time ($\mathcal{O}(n^n)$, with $n$ the number of nodes), since it involves checking all permutations. For this reason, the test is not practical and will not be considered further in this project.

### 1.2.3. Easy Region and Spectral Test

Finally, in the *easy region*, detection is computationally feasible whenever

$$1 - \beta \gtrsim \left(\frac{\sqrt{\log n}}{k} \vee \frac{1}{k}\right).$$

Here, the *spectral test* provides a polynomial-time alternative, with complexity $\mathcal{O}(n^2)$. Unlike the MLE test, which searches exhaustively through many permutations, the spectral test only analyzes the dominant eigenvalues of the adjacency matrix, making it much faster. This test will be used on artificial data to determine the accuracy of the Spectral test. The test statistic is the second-largest eigenvalue of the adjacency matrix:

$$T_2(A) = \lambda_2(A).$$

The reason eigenvalues are studied is that they contain rich information about the structure of a graph. The largest eigenvalue for both ER and WS networks can be found by multiplying the adjacency matrix by the all-ones vector $\mathbf{1} = (1,1,\ldots,1)^\top \in \mathbb{R}^n$. This shows that $\lambda_{\max}$ for both networks essentially reflects the average degree of the network. Since in our definitions both WS and ER networks have an average degree of order $k$, the largest eigenvalue is also of order $k$. Therefore, $\lambda_{\max}$ provides no additional information about whether the graph is random or small-world.

For an Erdős–Rényi graph with $n$ nodes and edge probability $\beta\frac{k}{n-1}$, the entries of the adjacency matrix $A$ satisfy

$$\mathbb{P}(A_{ij} = 1) = \beta\frac{k}{n-1}, \qquad \mathbb{P}(A_{ij} = 0) = 1 - \beta\frac{k}{n-1}, \quad (i \neq j),$$

and $A_{ii} = 0$ (no self-loops). Thus

$$\mathbb{E}[A_{ij}] = \beta\frac{k}{n-1}, \qquad \mathrm{Var}(A_{ij}) = \beta\frac{k}{n-1}\left(1 - \beta\frac{k}{n-1}\right).$$

The expectation of the adjacency matrix is

$$\mathbb{E}[A] = \beta\frac{k}{n-1}(J - I),$$

where $J = \mathbf{1}\mathbf{1}^\top$ and $I$ is the identity matrix. Since $J$ has rank one, $\mathbb{E}[A]$ has exactly one nonzero eigenvalue of order $k$, while the rest vanish. Using results from Wigner (1958), on random matrix theory, the remaining eigenvalues lie within a circle of radius

$$\sqrt{n \cdot \beta\frac{k}{n-1}\left(1 - \beta\frac{k}{n-1}\right)} \sim \sqrt{k},$$

for large $n$. Thus, apart from the top eigenvalue, all other eigenvalues of an ER graph are of order $\sqrt{k}$ and lie inside this circle, called the “bulk.”

—

For a Watts–Strogatz (WS) graph, however, the expectation has an additional contribution from the underlying ring lattice. In fact,

$$\mathbb{E}[A] \;=\; (1-\beta)(1-\beta\frac{k}{n-1})\,A_{\text{RRL}} \;+\; \beta\cdot\frac{k}{n-1}(J-I),$$

where $A_{\text{RRL}}$ is the adjacency matrix of the original ring lattice. The second term matches the ER expectation, but the first term is new: it encodes the lattice structure and therefore raises the rank of the expectation. Since both $A_{\text{RRL}}$ and $J-I$ are circulant and symmetric matrices, they share the same eigenvectors and can be diagonalized together Nordgren (2020); Strang (2017). This expression can thus be rewritten as

$$\mathbb{E}[A] = D\,B\,D^{\top},$$

where $D \in \mathbb{R}^{n\times n}$ is orthogonal and $B = \operatorname{diag}(\lambda_{max}, \lambda_2, \dots, \lambda_{n-1})$. As stated earlier, the eigenvalues of a regular ring lattice are:

$$\lambda_r\big(A_{\text{RRL}}\big) = 2\sum_{s=1}^{k/2}\cos\Big(\tfrac{2\pi rs}{n}\Big) \qquad (k \text{ even}).$$

For $J-I$,

$$\lambda_{max}(J-I) = n-1, \qquad \lambda_r(J-I) = -1 \;\; (r = 1,\dots,n-1).$$

This is because the eigenvalues of $J$ are $\lambda_{max}(J) = n$ and the rest are zeros. Subtracting $I$ shifts all eigenvalues by $-1$. Let $\gamma(\beta,k) = (1-\beta)\big(1-\beta\frac{k}{n-1}\big)$. Since $A_{\text{RRL}}$ and $J-I$ are diagonalizable in one orthogonal matrix $D$, Motzkin and Taussky's theory applies (Motzkin and Taussky (1955)):

$$\lambda\Big(\gamma(\beta,k)A_{\text{RRL}} + \beta\tfrac{k}{n-1}(J-I)\Big) = \gamma(\beta,k)\,\lambda(A_{\text{RRL}}) + \beta\tfrac{k}{n-1}\,\lambda(J-I).$$

Then the eigenvalues of $\mathbb{E}[A]$ are

$$\lambda_{max}(\mathbb{E}[A]) = \gamma(\beta,k)\,k+\beta\,k = k+k^2\left[\frac{\beta(\beta-1)}{n-1}\right], \qquad \lambda_r(\mathbb{E}[A]) = \gamma(\beta,k)\cdot 2\sum_{s=1}^{k/2}\cos\Big(\tfrac{2\pi rs}{n}\Big) - \beta\,\tfrac{k}{n-1}, \;\; r = 1,\dots,n-1.$$

Because of this higher-rank structure, additional eigenvalues appear outside the bulk. Indeed, as shown by Chauhan et al. (2009), the number of large eigenvalues corresponds to the number of clusters (or communities) present in the network. Every WS graph has at least one such cluster, so the second-largest eigenvalue $\lambda_2$ already contains the relevant information. Since:

$$|\cos\left(\tfrac{2\pi s}{n}\right)| \le 1,$$

by the triangle inequality:

$$\left|\sum_{s=1}^{k/2}\cos\left(\frac{2\pi s}{n}\right)\right| \;\le\; \sum_{s=1}^{k/2}\left|\cos\left(\frac{2\pi s}{n}\right)\right| \;\le\; \sum_{s=1}^{k/2}1 \;=\; \frac{k}{2}.$$

Substituting this in the equation for $\lambda_2$, $|\lambda_2|$ can be bounded by

$$|\lambda_2| \le k\gamma(\beta,k) + \beta\frac{k}{n-1}.$$

However, $\frac{k}{n-1} = o(1)$ (it goes to zero), meaning that $\gamma(\beta,k) = (1-\beta)\big(1-\beta\frac{k}{n-1}\big) = \mathcal{O}(1)o(1) = \mathcal{O}(1)$. Then

$$|\lambda_2| \le k\mathcal{O}(1) - o(1) = \mathcal{O}(k).$$

If $\lambda_2$ is asymptotically larger than order $\sqrt{k}$, this indicates the presence of small-world structure. The precise spectral test based on this principle is described below, with its parameters justified in Section 2.

**Spectral Test** The spectral test $\phi_2 : A \to \{0, 1\}$ is defined as

$$\phi_2(A) = \begin{cases} 1 & \text{if } T_2(A) \geq \sqrt{k} \vee \sqrt{\log n}, \\ 0 & \text{otherwise.} \end{cases}$$

When $\phi_2(A) = 1$, $H_0$ can be rejected and the graph is considered small-world. If $\phi_2(A) = 0$, the null cannot be rejected.

### 1.2.4. Size and Power

To evaluate the performance of the tests for finite $n$, we apply the test to simulated networks with known $n$, $k$, and $\beta$, allowing empirical estimates of size and power. We are interested in the power of the test. The power of a test is the probability of correctly rejecting the null hypothesis, also known as the complement of type II errors. The general definition is:

$$\theta \to \pi(\theta; K) = \mathbb{P}_\theta(X \in K),$$

where $\theta$ represents the true parameter values and $K$ is the region where $H_0$ gets rejected. In this thesis, $\theta = \beta$ and $K = \{\text{reject } H_0\}$. $H_0$: $\beta = 1$ and $H_1$: $\beta < 1$. The power function then becomes:

$$\pi(\beta) = \mathbb{P}_\beta(\text{reject } H_0).$$

Ideally, we want $\pi(1)$ to be small and $\pi(\beta < 1)$ to be large.

In addition, the size of the test is of interest. Which is the probability of falsely rejecting the null hypothesis, also called a type I error. More formally, the size of a statistical test is defined as follows:

$$\alpha = \sup_{\theta \in \Theta_0} \pi(\theta; K),$$

where $\pi$ is the power function. Again, the definition becomes:

$$\alpha = \pi(1) = \mathbb{P}_1(\text{reject } H_0).$$

A strong statistical test has power close to 1 when $H_1$ is true, and close to $\alpha$ when $H_0$ is true. However, the power and size of the test are intertwined: if the number of type I errors decreases, the number of type II errors increases. To compare the accuracy of two tests, it is common practice (see for example, Bijma et al. (2026)) to fix the $\alpha$ level and then compare the power. In this paper, $\alpha = 0.05$, unless stated otherwise.

While the spectral test is convergent for $n \to \infty$, according to Cai et al. (2017), there is no guarantee for its size for any $n$ value. This means that the number of type I errors cannot be contained and must be estimated via a Monte Carlo method: simulating many graphs under the null hypothesis (ER) and calibrating the test accordingly to estimate the size. This is possible for simulated data, because many graphs can be generated with certainty that $\beta = 1$, but with real data, this is not certain and this Monte Carlo method cannot be used. The spectral test will thus not work for real data.

## 1.3. Tracy-Widom

The second statistical test developed in this thesis is expected to have $\alpha$-level significance and is based on the Tracy–Widom distribution. To develop a test statistic using the TW distribution, it is necessary to estimate $k$ as it is part of the test statistic that will be developed in later sections. Estimating $\beta$ will also prove useful, because when the test rejects, we have an estimate to know how deterministic the network is.

### 1.3.1. Estimating the parameters

The parameters are estimated using the *maximum likelihood method*, a common method. Let $X$ be a random variable with probability density $p_\theta$ that depends on parameter $\theta \in \Theta$. For a fixed $x$, the function

$$\theta \to p_\theta(x)$$

is called the likelihood function. The maximum likelihood estimate of $\theta$ is the value $T(x) \in \Theta$ that maximizes the likelihood function $\theta \to p_\theta(x)$.

Often, the logarithm of $p_\theta$ is taken to reduce the number of calculations; this is called the log-likelihood function (Bijma et al. (2026)). For the constructed small-world graphs from Section 1.1.3, the probabilities of two nodes having an edge can be divided into two separate probabilities:

$$p_L = 1 - \beta - \beta^2 \frac{k}{n-1} \quad \text{and} \quad p_R = \beta \frac{k}{n-1}. \tag{1.2}$$

Here, $p_L$ is the probability of two neighboring nodes being connected, as they would in a regular ring lattice. $1-\beta$ is the probability that the edge stays, and $\beta^2 \frac{k}{n-1}$ is the probability it is re-added during reconstruction. $p_R$ is the probability of two non-neighbor nodes having an edge, representing the random structure. For the MLE, the total number of lattice edges and non-lattice edges is compared to the number of lattice and non-lattice edges actually present in the graph. The total number of possible edges is: $M_L = \frac{nk}{2}$, since each node has $k$ edges, divided by two to avoid double counting. $M_R = \binom{n}{2} - M_L = \frac{n(n-1)}{2}$, where $\binom{n}{2} = \frac{n(n-1)}{2}$ is the total number of possible edges in a graph with $n$ nodes. Both numbers of edges are represented by a binomial distribution:

$$m_L \sim \text{Bin}(M_L, p_L) \qquad m_R \sim \text{Bin}(M_R, p_R)$$

The likelihood function is the probability of observing $m_L$ and $m_R$. Since we are estimating two parameters, namely $\beta \in [0,1]$ and $k$ (an even integer), the likelihood function is a function of $\beta$ iterated over all possible $k$ values:

$$L(b \mid k) = \binom{M_L}{m_L} p_L^{m_L} (1-p_L)^{M_L - m_L} \cdot \binom{M_R}{m_R} p_R^{m_R} (1-p_R)^{M_R - m_R},$$

Taking the log-likelihood gives

$$\ell(b \mid k) = m_L \log(p_L) + (M_L - m_L)\log(1-p_L) + m_R \log(p_R) + (M_R - m_R)\log(1-p_R).$$

where the binomial constants have been omitted, as they are not dependent on $\beta$.

This log-likelihood function is iterated over $k$ and finds the parameter $\hat{b}(k)$ for which it is maximized. Of all the pairs $(\hat{b}(k), k)$, the MLE estimators are

$$(\hat{b}, \hat{k}) = \arg \max_{k \in \{2,4,\ldots,n-2\}} \ell(b(k) \mid k).$$

The accuracy of the estimators is determined using the Mean Squared Error (MSE), and the average distance from the true value is calculated using the Root Mean Squared Error (RMSE). The MSE evaluates how close the estimated parameters are to their true values by squaring and averaging the estimation errors over multiple realizations. This measure penalizes large deviations more heavily, providing an accurate indicator of estimation quality. Taking the square root of the MSE yields the RMSE, which translates the error back into the same scale as the original parameter. The RMSE can therefore be interpreted as the " quadratic average distance" between the estimator and the true value, making it an intuitive measure of practical accuracy. Knowing that the estimations are accurate is important; if the error is too large then the test statistic that will be developed later will have too much 'noise' and will be unusable. They are defined (in Bijma et al. (2026) and Hodson (2022)) as follows: $MSE(\beta; T) = \text{var}_\beta(T) + (\mathbb{E}_\beta[T] - \beta)^2$, where $T$ is the estimator. As will be clear in Section 3, the second term of the equation, called the bias, is zero in this case. $RMSE = \sqrt{MSE}$ .

As will be seen in Section 3 and A, this MLE estimation is very accurate but has a drawback: the estimation of $\beta$ is inaccurate when the adjacency matrix undergoes random permutation, this is because the algorithm can count if a pair is a neighbor or not, based on the original ring lattice structure. Random permutation destroys this structure so it cannot compare the adjacency matrix to any mask. For artificial data this method works, but most real networks are permuted, meaning the MLE is not a good estimator of the parameters. In Section 5 another method is presented that works under permutation.

### 1.3.2. Tracy-Widom distribution

The Tracy–Widom distribution can be defined as follows:

$$F_\tau(s) = \exp\left(-\int_s^\infty (x-s)q^2(x)\,dx\right),$$

where $q(x)$ solves the Painlevé II differential equation

$$q''(x) = xq(x) + 2q^3(x),$$

Here, $\tau = \{1, 2, 4\}$, but in the case of this thesis $\tau = 1$. This corresponds to the **GOE**, Gaussian Orthogonal Ensemble, which is a model for random symmetric matrices whose entries are independent normal random variables (according to Tracy and Widom (1994) and Cugliandolo (2019)).

According to Lee and Schnelli (2016), in the limit, the second-largest eigenvalue of an ER matrix follows the Tracy–Widom distribution:

$$\mathbb{P}\left(n^{2/3}\,(\lambda_2 - L - a) \le s\right) \xrightarrow[n\to\infty]{} F_1(s).$$

with $L = 2 + \frac{1}{d}$, $d$ the average degree of the matrix, and $a = \sqrt{\frac{p}{n(1-p)}}$ where $p = \frac{\beta k}{n-1}$, whenever $\lambda_2$ is scaled by $\frac{1}{\sqrt{np(1-p)}}$. As will be seen in Section 3, for both a small-world and a random graph, this $T$ will converge to a scaled and shifted TW distribution. However, we expect that

$$\mathbb{P}\left(n^{2/3}\,(\lambda_2 - L - a_0) \le s\right)$$

with $a_0 = \sqrt{\frac{p}{n(1-p)}}$, but with $p = \frac{k}{n-1}$, will not converge to a TW distribution when the graph is small-world. This makes $T_0 = n^{\frac{2}{3}}(\lambda_2 - L - a_0)$ a usable test statistic.

The null hypothesis for this test is thus again that the graph is completely random, in which case $T_0$ converges to TW, and the alternative hypothesis is that the graph is small-world and $T_0$ has a different distribution. The accuracy of this test can again be determined using artificially simulated graphs with a known $\beta$ value.

## 1.4. Real-World Data

It is interesting to examine how these tests perform on real data. Therefore, both tests are applied to a graph representing the characters from the novel *Les Misérables*, as used by Cai et al. (2017). Each node corresponds to a character, and an edge between two nodes indicates that the characters interacted at some point in the novel. A visualization of the network is shown in Figure 1.5. In this network $n = 77$ and $k = 8$, which means the graph is quite sparse, but fits the necessary conditions for WS. $\beta$ can be estimated using the MLE since there is no permutation on this graph. The hypothesis is that this graph exhibits small-world properties, as suggested by Cai et al. (2017). While this analysis is insightful, it does not provide information about the accuracy of the tests. One can, in addition, estimate the $\beta$ value of said graph to know how much structured the graph contains.

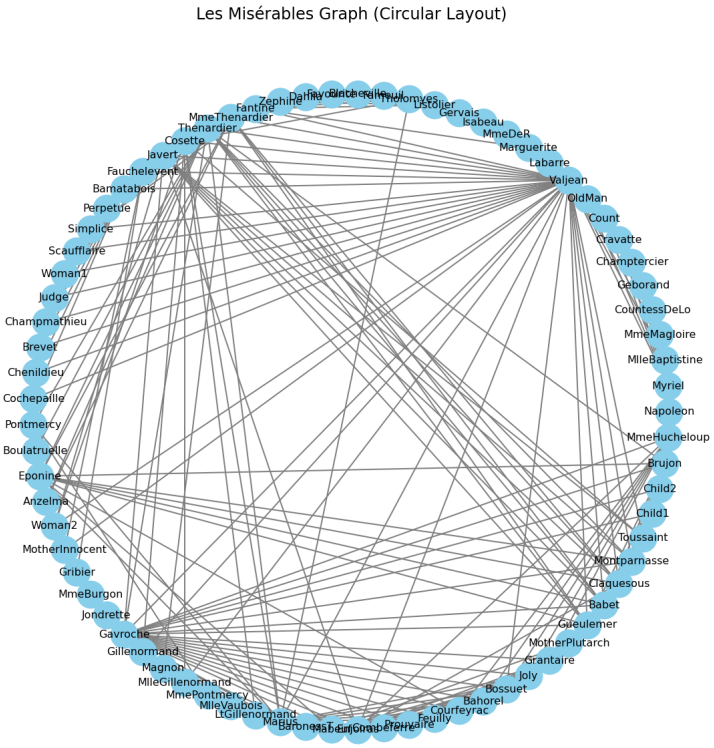


Figure 1.5: Network of character interactions in *Les Misérables*.

# 2

# Method

All experiments in this thesis are implemented in Python. The relevant code is provided in the Appendix A.

The first step is to develop an algorithm that generates small-world graphs with user-defined parameters $n$, $k$, and $\beta$. This ensures that the true structure of the graph is known, making it possible to evaluate the accuracy of the detection methods. The second step generates as many graphs as needed for testing. This allows for large-scale simulations and more statistically reliable results. Each generated graph is then converted into its corresponding adjacency matrix. The result is a large collection of adjacency matrices, on which the tests are performed.

## 2.1. Spectral test

### 2.1.1. Spectral Test and Monte Carlo

The spectral test, as defined in the introduction, is implemented in Python as a function that iterates over all generated adjacency matrices and computes the second-largest eigenvalue $\lambda_2$ for each. These eigenvalues are then compared against the threshold

$$c \cdot \left(\sqrt{k} \vee \sqrt{\log n}\right),$$

where $c$ is a constant determined using the Monte Carlo method. Specifically, $2000$ Erdős–Rényi (ER) graphs are generated, and for each graph, the ratio $R$ is computed:

$$R = \frac{\lambda_2}{\sqrt{k} \vee \sqrt{\log n}}.$$

The constant $c$ is chosen as the $95$th quantile of these ratios. This choice is validated by performing the same Monte Carlo experiment on WS graphs with varying values of $\beta$. For example, when $n = 100$ and $k = 10$, the histogram shows that the values of $\lambda_2$ for WS and ER graphs overlap only minimally; indicating that $c$ is a suitable constant to ensure asymptotic separation. This result is shown in Figure 2.1.

It is also instructive to consider $\beta = 0.75$, which lies above the ”easy region” for this choice of $n$ and $k$. In this case, the second largest eigenvalues of the WS and ER graphs overlap significantly, making it impossible to reliably distinguish the two models using this method. This result is shown in Figure 2.2.

Whenever

$$\lambda_2 > c \cdot \left(\sqrt{k} \vee \sqrt{\log n}\right),$$

the function assigns the value $1$ to the graph, meaning that the null hypothesis $H_0$ is rejected. Otherwise, it assigns the value $0$. This yields a list of outcomes, where $1$ denotes rejection of $H_0$ and $0$ denotes failure to reject $H_0$. Following Watts and Strogatz (1998), the condition $\ln n \ll k \ll n$ must be met to guarantee that the rewired WS graph remains connected. To explore how different ratios of $k$ and $n$

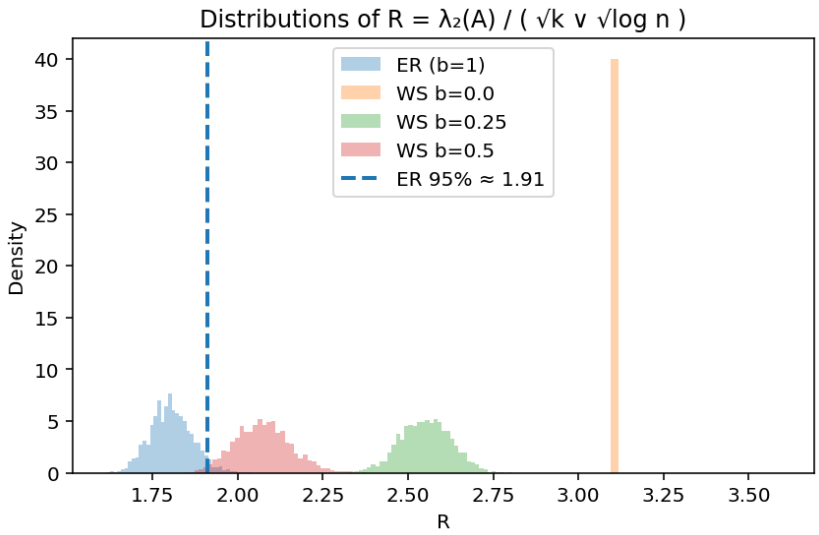


Figure 2.1: Histogram of $\frac{\lambda_2}{\sqrt{k} \vee \sqrt{\log n}}$ values for ER and WS graphs. $\beta$ takes values 0, 0.25, and 0.5. The 95th quantile line of the ER graphs is shown.

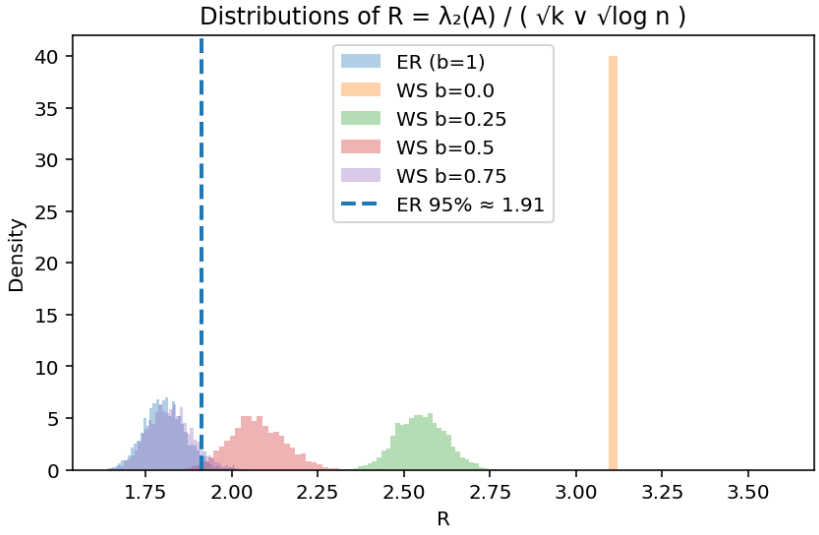


Figure 2.2: Histogram of $\frac{\lambda_2}{\sqrt{k} \vee \sqrt{\log n}}$ values for ER and WS graphs. $\beta$ takes values 0, 0.25, 0.5, and 0.75. The 95th quantile line of the ER graphs is shown.

affect performance, and to test the boundaries of the spectral test, experiments are conducted for two ratios:

$$\frac{k}{n} = 0.1, \quad \frac{k}{n} = 0.5.$$

### 2.1.2. Size and Power

The *size* of the test is estimated by applying it to many ER graphs and calculating the proportion of times the null hypothesis is incorrectly rejected. This provides an empirical estimate of the type I error rate. The *power* of the test is estimated by applying it to many WS graphs and calculating the proportion of times the graphs are correctly classified as WS (i.e., $H_0$ is rejected). This provides an empirical estimate of the complement of the type II error (i.e., $1$ – type II error).

## 2.2. Tracy-Widom

### 2.2.1. Parameter estimation

The estimation of the parameters is carried out using several functions in Python. First, a ”mask” is constructed, which is a regular ring lattice graph with entries set to ”True” for all neighbor edges. The code compares the given adjacency matrix with the mask and then counts which edges are neighbors (and how many) and which edges are randomly added (and how many). These values are then used in the log-likelihood function to determine the best estimates for $\beta$ and $k$. This procedure is repeated for 1000 adjacency matrices with $n = 100$ and $k = 10$, where $\beta = 1, 0.75, 0.5,$ and $0.25$. The results are displayed in histograms, with the mean indicated.

### 2.2.2. Tracy-Widom

All relevant scripts to calculate the test statistic $T$ are implemented in Python. The test uses the same controlled adjacency matrices as in the spectral test. It iterates over each matrix, computes $\lambda_2$, and then calculates $T$. All values are plotted in a histogram, with a scaled Tracy-Widom distribution shown as a red line. This is done for both a set of ER graphs and a set of WS graphs. Each histogram is accompanied by a QQ-plot to evaluate how well the data matches the ER distribution. The p-value and the KS-value are reported too quantitatively describe the match.

For the actual test, the test statistic $T_0$ is calculated. For each adjacency matrix, it checks whether this value is larger than the cutoff value $t$, which in this case is the 95th quantile of the Tracy-Widom distribution. If so, the test assigns the value $1$ to the matrix, meaning the null hypothesis is rejected; otherwise, it assigns $0$. Finally, the power and size are plotted to assess the accuracy of the test.

## 2.3. Real-world data

The data are gathered from a built-in function in Python, which is turned into an adjacency matrix consisting of zeros and ones. In this format, the network can be evaluated by the tests and returns a 1 if it rejects and a 0 if it doesn’t. The parameter $\beta$ is estimated using maximum likelihood estimation.

# 3

# Results

## 3.1. Spectral Test

The power of the spectral test is shown for different ratios of $k$ and $n$ as the rewiring probability $\beta$ increases. Throughout this section, $n$ is fixed at $100$.

For both $\frac{k}{n} = 0.1$ and $\frac{k}{n} = 0.5$, which satisfy the condition set by Watts and Strogatz (1998), the power is close to $1$ for roughly $\beta < 0.5$ and $\beta < 0.65$, respectively. Beyond these thresholds, the power drops rapidly to near zero. See Figure 3.1.

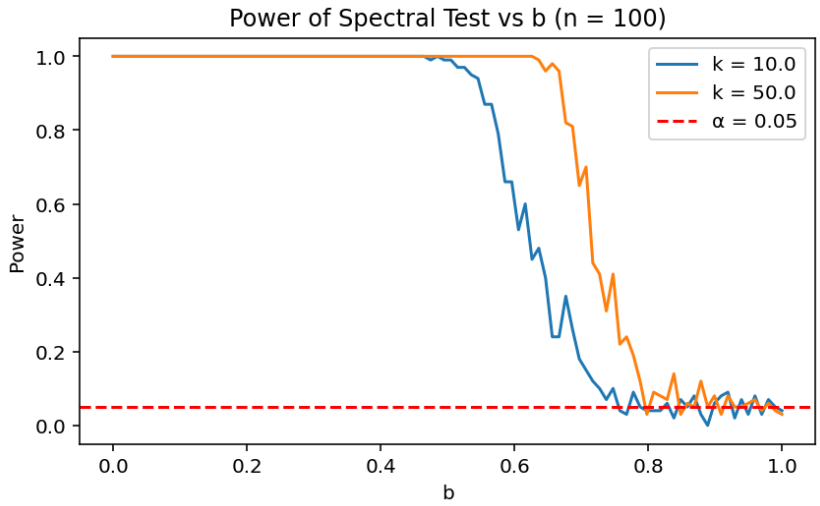


Figure 3.1: The size and power for different $\frac{k}{n}$ values are plotted as $\beta$ increases.

Figure 3.2 shows the power of the spectral test for different values of $\beta$ as $n$ increases.

The power of the spectral test is higher for smaller $\beta$, and for each fixed $\beta$, it increases as $n$ grows, except for $\beta = 0.75$ where the power remains at the value of $\alpha$.

## 3.2. Tracy Widom

### 3.2.1. Parameter Estimation

The estimation for $k$ can be seen in the histogram in Figure 3.3. There are some estimates at $k = 8$ and $k = 12$, but most values are correctly estimated at $k = 10$. The MSE is at most $0.08$ and the RMSE is at most $0.3$. The estimation for $\beta = 0.25$ can be seen in Figure 3.4. The MSE is at most $0.0004$, and the RMSE is at most $0.02$. The mean of the estimated values is exactly 0.250, with the data tightly concentrated around this value. Additional plots for different $\beta$ values can be found in the Appendix A.

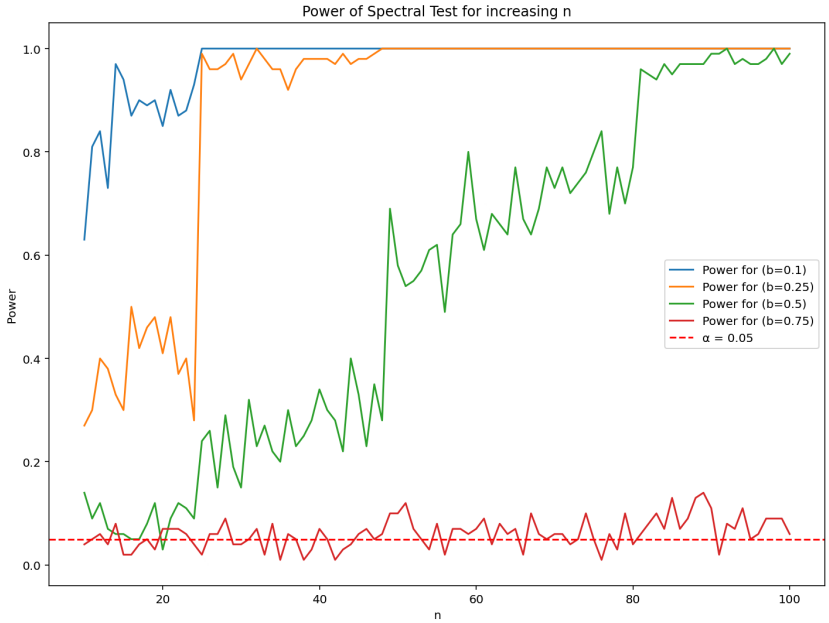


Figure 3.2: The size and power of the Spectral test for different $\beta$ values are plotted as $n$ increases.

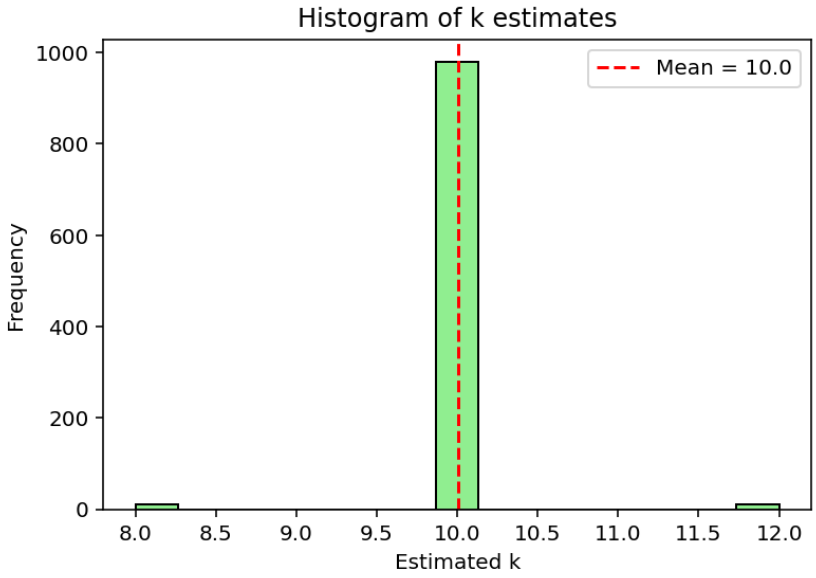


Figure 3.3: Histogram of estimated $k$ values for $n = 100$ and $k = 10$.

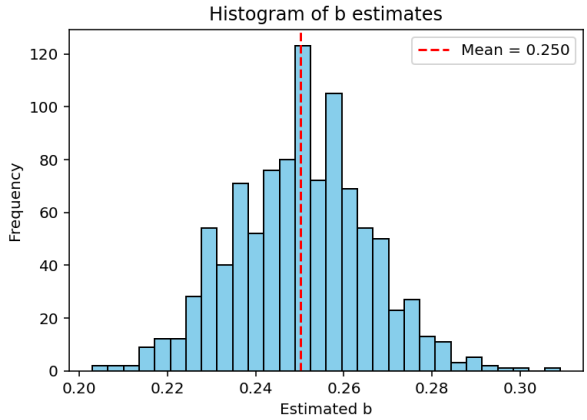


Figure 3.4: Estimation for $\beta$ for the adjacency matrix with $n = 100$, $k = 10$, and $\beta = 0.25$.

### 3.2.2. Tracy-Widom

**ER networks ($\beta = 1$)** We first examine the test statistic $T_0$ for ER graphs and plot the histogram of the values together with the QQ-plot. The histogram can be seen in Figure 3.5. The red line is the scaled and shifted theoretical TW distribution, to match the mean and variance of the empirical data (more on this in Section 5). The QQ-plot, on the other hand, is shown to visualize the overlap between the theoretical and empirical distribution, see Figure 3.6. The more overlap, the better the fit. The corresponding KS goodness of fit test is included. This is defined as $D = \sup_x |T_0(x) - TW(x)|$, the maximum distance between the empirical data and the Tracy-Widom fit, see Zeimbekakis (2022). Small $D$ indicates a good fit. In addition, the corresponding p-value is included. The p-value is the probability, under the null hypothesis that the data follow the Tracy–Widom distribution, of observing a KS statistic at least as large as the one computed from the sample. A large p-value therefore indicates that $H_0$ cannot be rejected. It is evident that the empirical data aligns well with the scaled and shifted TW distribution when $\beta = 1$.

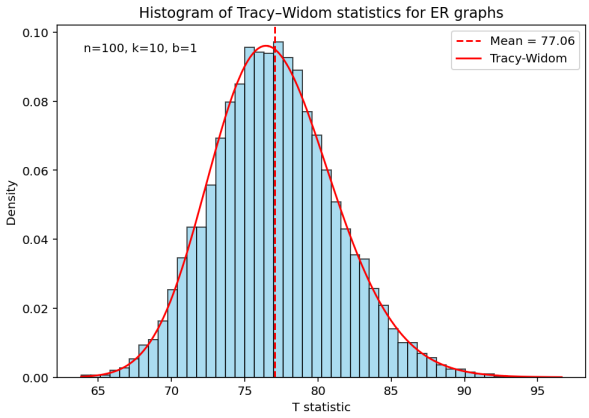


Figure 3.5: Histogram of $T$ for $\beta = 1$, $n = 100$, and $k = 10$. The red line represents the scaled TW distribution.

**WS networks ($\beta < 1$)** Figures 3.7 and 3.8 show similar plots, but for the WS graphs. Once again, the empirical distribution follows the theoretical TW distribution when $\beta = 0.2$. Appendix A also includes a histogram and QQ-plot for $\beta = 0.8$.

**Power** The power of the TW test is plotted in two dimensions: (i) fixing $\frac{k}{n}$ for increasing $\beta$ and (ii) fixing several $\beta$ values for increasing $n$. Firstly, one can see the power for increasing $\beta$ with $\frac{k}{n} = 0.1$ and $\frac{k}{n} = 0.5$. For both ratios, the power approaches 1 for $\beta < 0.5$ and $\beta < 0.65$, respectively, after

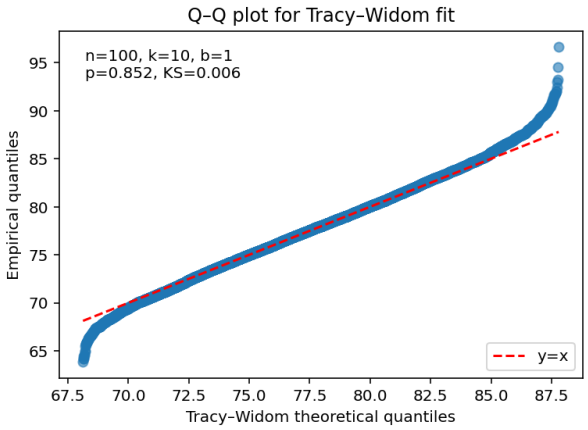


Figure 3.6: QQ-plot illustrating the agreement between the TW distribution and $T$ for $\beta = 1$, $n = 100$, and $k = 10$. The corresponding $p$- and KS-values are reported.

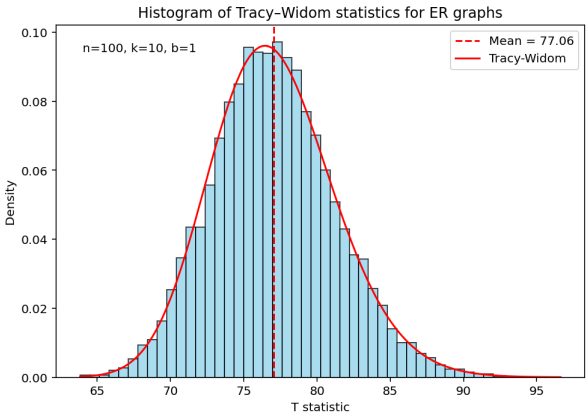


Figure 3.7: Histogram of $T$ for $\beta = 0.2$, $n = 100$, and $k = 10$. The red line represents the scaled TW distribution.

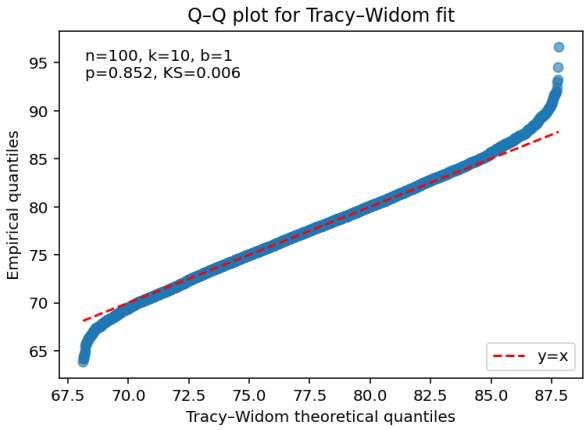


Figure 3.8: QQ-plot illustrating the overlap between the TW distribution and $T$ for $\beta = 0.2$, $n = 100$, and $k = 10$. The corresponding $p$- and KS-values are reported.

which it drops back to the $\alpha$ level around $\beta \approx 0.75$ (see Figure 3.1). Second, the power of the Tracy–

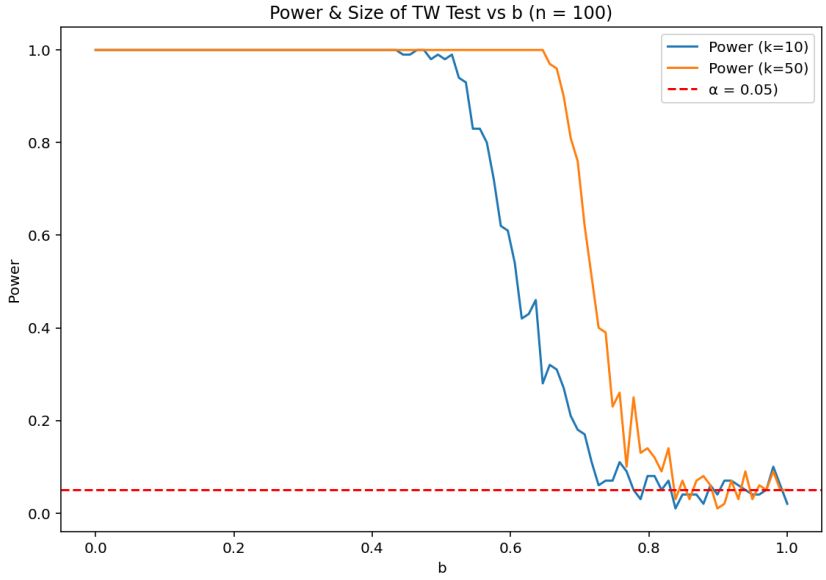


Figure 3.9: The size and power of the TW test for different $\frac{k}{n}$ values as $\beta$ increases.

Widom test is further plotted as a function of $n$ for various $\beta$ values. The test begins to gain power once $n > 25$. For $25 < n < 45$, it achieves moderate power when $\beta = 0.1$ and $\beta = 0.25$, although still lower than the spectral test. For larger $n$, the power increases further. However, for these $\beta$ values, the power reaches 1 around $n = 45$. As with the spectral test, it exhibits no power for $\beta = 0.75$.

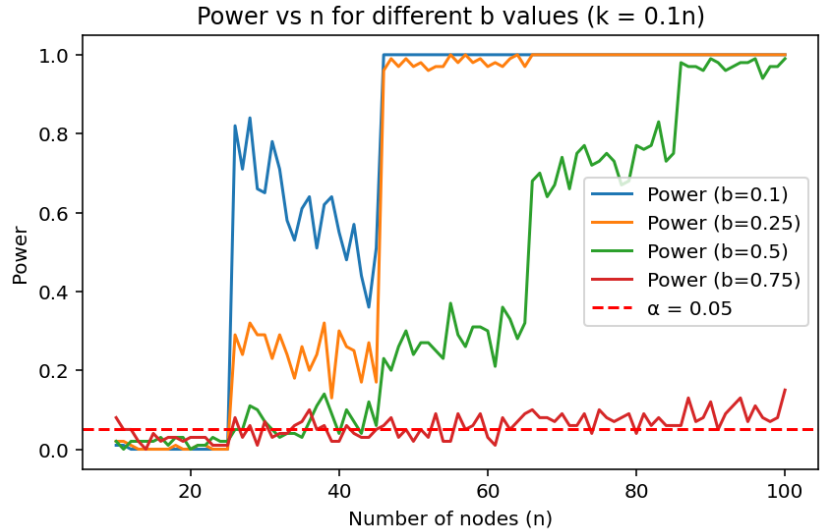


Figure 3.10: The size and power of the TW test for different $\beta$ values as $n$ increases.

**Comparing the Spectral Test with the TW Test** To directly compare the performance of both tests, their power functions are plotted together in Figures 3.11 and 3.12. As can be seen from Figure 3.11, both tests have very similar power for $\beta$ values ranging from 0 to 1, where they both start dropping around $\beta = 0.5$ and lose power from $\beta > 0.8$. Figure 3.12 plots the power for different $\beta$ values as $n$ increases. Once again, the powers behave very similarly. Neither test has power for $\beta = 0.75$, while for the other $\beta$ values, the test has power for $n$ larger than 25 and $\beta = 0.25$ and $\beta = 0.1$ have power 1 for $n$ greater than 45.

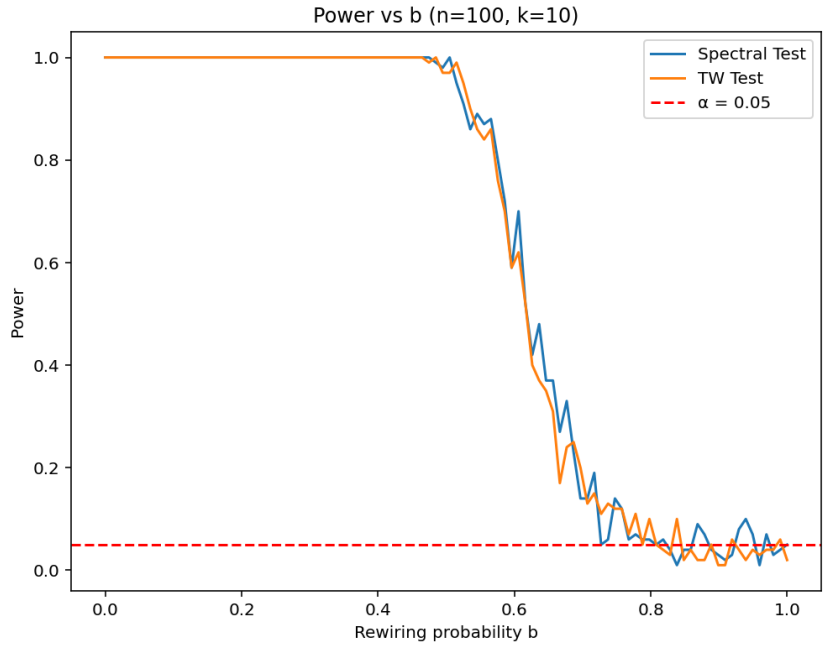


Figure 3.11: The size and power of both tests for $\frac{k}{n} = 0.1$ as $\beta$ increases.

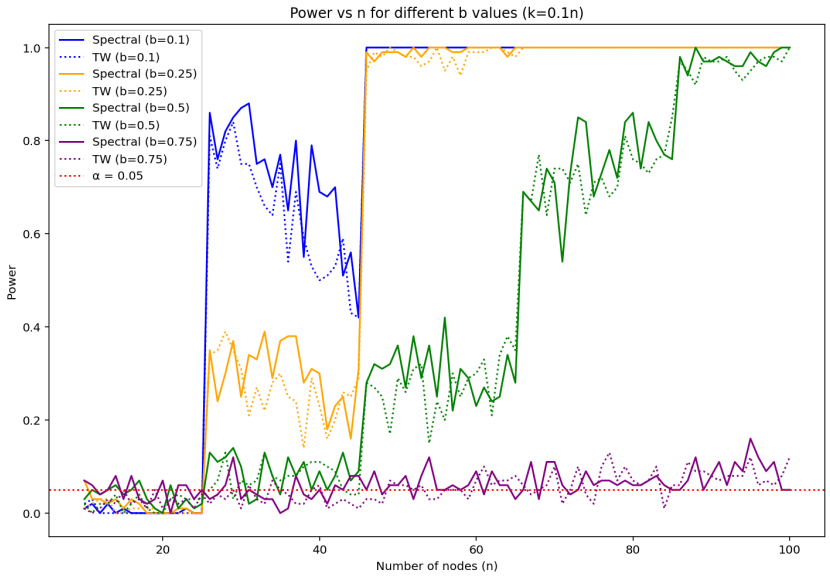


Figure 3.12: The size and power of both tests for various $\beta$ values as $n$ increases.

## 3.3. Real-world data

Both tests return the value 1, meaning that in both cases the null hypothesis—that the graph is an Erdős–Rényi graph—is rejected. This indicates that the network contains too much deterministic structure to be explained by a purely random model.

To quantify this structure, the parameter $\beta$ was estimated using the maximum likelihood method described in Section 1. The estimate $\hat{\beta} = 0.65$ suggests a moderate amount of structure: in the Watts–Strogatz paradigm, such a value corresponds to a small-world network, with substantially higher clustering than an ER graph while still retaining enough rewiring to produce short path lengths.

Together, the hypothesis test results and the estimated rewiring probability provide consistent evidence that the *Les Misérables* network exhibits meaningful small-world structure rather than being adequately described by an Erdős–Rényi model.

# 4

# Conclusion

## 4.1. Spectral Test

The power behaves as expected, provided that the ratio of $n$ and $k$ remains within the predetermined bound set by Watts and Strogatz (1998), meaning the 'easy region'. When this condition is satisfied (i.e., for $\frac{k}{n} \leq 0.5$), the power is close to $1$ when $\beta$ is small, and it decreases rapidly once $\beta$ passes a certain threshold. This occurs because $\beta$ leaves the "easy region,"; here it becomes increasingly difficult to distinguish between a small-world graph and an Erdős–Rényi graph. Consequently, the number of cases in which the test fails to reject $H_0$ increases. It can also be observed that the power of the spectral test occasionally falls below the significance level of the test ($\alpha = 0.05$). This indicates that the test has essentially no power and therefore cannot be regarded as a valid level-$\alpha$ test. Finally, when $\beta$ is fixed but $n$ increases, the power also increases. This is expected, since the estimation error decreases as $n \to \infty$. The power for $\beta = 0.75$ remains low because this value lies outside the region in which the spectral test is effective. The "impossible region" occurs when $\beta$ is substantially larger than $0.54$. Thus, it is unsurprising that in Figure 3.1 the test shows no power for $\beta > 0.8$, and in Figure 3.2 it also shows no power for $\beta = 0.75$.

## 4.2. Tracy–Widom

The low MSE and RMSE confirm that the MLE accurately estimate $k$. One reason for this is that the Python code forces $k$ to be an even integer, rounding every value between $[9.1, 11]$ to $10$. This ensures a precise estimate, as nearly all estimated $k$ values equal $10$. The estimation for $\beta$ can likewise be considered accurate.

Furthermore, as seen in the combined plots for the spectral test and the TW test, both exhibit similar power. However, the TW test has a guaranteed $\alpha$-level significance, which must be set prior to testing (here, $\alpha = 0.05$). According to Lehmann and Romano (2005) and Morris and Elston (2012), the power—and thus the accuracy—of tests can only be meaningfully compared when they share the same $\alpha$ level. A test with a higher size is not more accurate, merely more liberal. Therefore, a test that maintains comparable power while guaranteeing $\alpha = 0.05$ is preferable. This implies that the TW test developed in this thesis is more favourable than the spectral test.

# 5

# Discussion

In this thesis two statistical tests were implemented using Python scripts to evaluate how accurately a small-world structure can be distinguished from and Erdős–Rényi graph. The first is the Spectral test introduced by Cai et al. (2017), and the other is a new test designed in this thesis based on the TW distribution. The performance of both methods could be assessed reliably by artificially simulating the data, ensuring the parameters were certain. Both tests exhibit similar power; however, the TW test maintains a controlled significance level $\alpha$, whereas the spectral test does not. Although these results are promising, several important considerations must be mentioned regarding the scope and limitations of the research.

## 5.1. Parameter validation

For the experiments, it was always assumed that $n = 100$, mostly because a larger order for $n$ was computationally too demanding for my laptop. However, most of the theories discussed are asymptotic, meaning that larger $n$ should yield better results. For example larger $n$ means a stronger Tracy–Widom fit and fewer type I and type II errors. Most tests have also used $k = 10$, which fits the criteria set by Cai et al. (2017) and Tracy and Widom (1994), but there are more values of $k$ that satisfy these requirements and may yield similar or different results. A further step in designing a test to accurately distinguish between WS and ER graphs could therefore be to test different ratios of $n$ and $k$ and to increase $n$.

## 5.2. Method of Moments estimation

Another method considered to estimate the parameters $k$ and $\beta$ was using the following equations:

$$\lambda_{max}(\mathbb{E}[A]) = \gamma(\beta,k)\,k + \beta\,k = k + k^2\left[\frac{\beta(\beta-1)}{n-1}\right], \qquad \lambda_2(\mathbb{E}[A]) = \gamma(\beta,k)\cdot 2\sum_{s=1}^{k/2}\cos\left(\frac{2\pi s}{n}\right) - \beta\,\frac{k}{n-1}.$$

and solving for $k$ and $\beta$. This was considered because the eigenvalues of a matrix remain the same after random permutation, which is the case for real-world graphs. However, this is not an ideal method as it only works in the easy region. When $\beta$ lies in the hard region:

$$1-\beta \gtrsim \left(\sqrt{\frac{\log n}{n}} \vee \frac{\log n}{k}\right),$$

(which for $n = 100$ and $k = 10$ is approximately $\beta \approx 0.5$,) the term behaves like $\frac{\log n}{k}$. That means that for

$$\lambda_2(\mathbb{E}[A]) = \gamma(\beta,k)\cdot 2\sum_{s=1}^{k/2}\cos\left(\frac{2\pi s}{n}\right) - \beta\,\frac{k}{n-1},$$

with $\gamma(\beta,k) = (1-\beta)\left(1-\beta\frac{k}{n-1}\right)$, we have $1-\beta \sim \frac{\log(n)}{k}$ and since $\beta\frac{k}{n-1}$ goes to zero,

$$\lambda_2(\mathbb{E}[A]) = \mathcal{O}\left(\frac{\log(n)}{k}\right)\mathcal{O}(k) + \sigma(1) = \mathcal{O}(\log(n)).$$

This is much smaller than in the easy region, where $\gamma \approx 1$ and

$$\lambda_2(\mathbb{E}[A]) = \mathcal{O}(k)$$

because of the condition set in Section 1: $\log(n) \ll k \ll n$. Thus, in the hard region, $\lambda_2$ is too small to provide information about $\beta$. Secondly, $\lambda_{max}$ (the maximum eigenvalue of the adjacency matrix) is governed by $\frac{\beta(\beta-1)}{n-1}$, which is a U-shaped function centered around $0.5$. This means that each eigenvalue corresponds to two possible $\beta$ values. When $\lambda_2$ contains no additional information, $\beta$ cannot be uniquely determined.

For any value of $\beta$, we observe that $\lambda_{max} \approx k$. This is because $\lambda_{max} = k(1+\beta(\beta-1)[\frac{k}{n-1}]) = k[1+\mathcal{O}(1)\sigma(1)] \approx k$. This is especially true if $\lambda_{max}$ is forced to be even and rounded. From this it is possible to solve the equation for $\lambda_2$ using $k = round(\lambda_{max})$ when in the easy region. If the graph is sparse enough (meaning that $\frac{k}{n} \to 0$), $\beta\frac{k}{n-1}$ is very small and becomes negligible. In this case $\beta \approx 1 - \frac{\lambda_s}{2\sum_{s=1}^{k/2}\cos(\frac{2\pi s}{n})}$, which is straightforward to solve. This method-of-moments approach is only possible in the easy region for sparse graphs, whereas the maximum likelihood estimation works for all values of $\beta$. However, the MLE does not work under permutation, while the MoM does. In Section A, two histograms are presented, showing the estimates of $k$ and $\beta$ using the method of moments (see Figure A.6 and A.7).

## 5.3. Rescaled TW distribution

As mentioned in Section 3, the histograms of the test statistic $T_0$ have been fitted with a *scaled and shifted* Tracy-Widom distribution. This is because a standard Tracy-Widom distribution has mean $\mu = -1.2065$ and standard deviation $\sigma = 1.6078$, as found by Bornemann (2010). The mean and variance found in Section 3, however, are significantly different, so in order to get a close fit, the theoretical TW distribution had to be rescaled and shifted using a procedure from Bijma et al. (2026): Let $X \sim TW$, then $Y = aX + b$, with $a = \frac{\sigma(T_0)}{\sigma}$ and $b = \mu(T_0) - a\mu$, where $\mu(T_0)$ and $\sigma(T_0)$ are the mean and standard deviation of the empirical data, respectively. $Y$ is then the rescaled TW distribution that fits the observations.

The reason this rescaling and shifting was necessary is because the empirical values do not converge to the standard Tracy-Widom distribution. One reason is that Lee and Schnelli (2016) requires that $\lambda_2$ be scaled by $\frac{1}{\sqrt{np(1-p)}}$, which was not done when calculating $T_0$. However, even when this is done, the empirical distribution still does not match TW well and this could have several reasons: First, the graph might not be sparse enough, which changes the shifting constant $L$. Second $\beta$ and $k$ were estimated, which could change the data too much to converge to a standard TW distribution. In addition, the constant $a_0$ may not be appropriate for this specific graph model, giving a wrong shift to $T_0$. Lastly, $n$ may not be sufficiently large, as according to Lee and Schnelli (2016), $T_0$ converges for $n \to \infty$. The precise cause of the discrepancy remains unclear.

## 5.4. Analytical power

Another reason to estimate $\beta$ is to calculate the analytical or theoretical power. This has the same mathematical definition as in Section 1, but instead of running the tests and taking the average number of correctly rejected $H_0$, it is possible to predict what the power should look like. Predicting it correctly can be very valuable, as it gives a good estimate of how well the test can ever perform. Using the test statistics:

$$T = n^{2/3}\left(\lambda_2 - L - a\right) \qquad T_0 = n^{2/3}\left(\lambda_2 - L - a_0\right).$$

with $T_0$ under $H_0$. We can develop the power as follows: Let $q_{\mathrm{TW},1-\alpha}$ be the $(1-\alpha)$-quantile of the Tracy–Widom law. In our case $\alpha = 0.05$. The test rejects the null when $T > q_{\mathrm{TW},1-\alpha}$. Since the power

is the probability of correctly rejecting the null hypothesis, we have:

$$\text{Power} \;=\; \mathbb{P}\big(T > q_{\mathrm{TW},1-\alpha} \,\big|\, H_1\big) \;=\; 1 - \mathbb{P}\big(T \le q_{\mathrm{TW},1-\alpha} \,\big|\, H_1\big).$$

$$\text{Under } H_1: \quad T_1 \;=\; n^{2/3}\big(\lambda_2 - L - a_1\big)$$

and we know that

$$T_1 \;=\; n^{2/3}\big(\lambda_2 - L - a_1\big) \;+\; n^{2/3}(a_1 - a_0),$$

which follows the Tracy–Widom distribution.

$$\Rightarrow \quad \mathbb{P}\big(T \le q_{\mathrm{TW},1-\alpha} \,\big|\, H_1\big) = \mathbb{P}\Big(T - n^{2/3}\big(a_0 - a_1\big) \;\le\; q_{\mathrm{TW},1-\alpha} - n^{2/3}(a_0 - a_1) \;\Big|\; H_1\Big).$$

And we know that

$$T - n^{2/3}\big(a_0 - a_1\big) \;=\; n^{2/3}\left(\lambda_2 - L - a_0\right) \;=\; T_0,$$

which has the Tracy–Widom distribution.

$$\text{Hence,} \quad \text{Power} \;=\; 1 - F_{\mathrm{TW}}\Big(q_{\mathrm{TW},1-\alpha} - n^{2/3}(a_0 - a_1)\Big),$$

where $F_X(t) = \mathbb{P}(X \le t)$. This theoretical power should be similar to the empirical power discussed in Section 3, but it remains very small. This could be due to the fact that $n^{2/3}(a_0 - a_1)$ is quite small, and for $n = 100$ this shift is too small to detect. In addition, a reason for this discrepancy could be that the empirical data does not converge to a standard TW distribution for reasons explained in the previous sections. This means that if $T_0$ does follow a standard TW distribution, the analytical power can be very useful.

# A Appendix

## A.1. Python code

The relevant code is included here, including the spectral test, the estimation procedures for $k$ and $\beta$, the TW test, and the scripts that generate all the plots.

### A.1.1. Spectral Test

```
1
2
3  import networkx as nx
4  import numpy as np
5  import random
6  import matplotlib.pyplot as plt
7
8
9
10 #VARIABLES:
11 n = 100                                                          #Number of nodes.
12 k = int(np.sqrt(n) )                                            #Number of
       connected neighbors.
13 k += k % 2                                                       #Force k to be
       even.
14 h = 100                                                         #Number of graphs
       to generate.
15 max_b = 1 - 1.5 * max(1/np.sqrt(k), np.sqrt(np.log(n))/k)       #Max value b can
       have for a large power.
16 b = 0.1                                                         #Actual b used in
       experiments.
17
18
19 #FUNCTIONS
20 def ArtSimGraph(n,k,b):  #Generate an WS or ER graph.
21
22     G = nx.Graph()
23     G.add_nodes_from(range(n))
24
25     # Step 1: Connect each node to k/2 neighbors on each side (ring lattice)
26     for node in range(n):
27         for i in range(1, int((k/2)) + 1):
28             neighbor = (node + i) % n
29             G.add_edge(node, neighbor)
30
31
32       # Step 2: Remove edges with probability b
33     edges_to_remove = []
34     for u, v in G.edges():
35         if random.random() < b:
36             edges_to_remove.append((u, v))
37     G.remove_edges_from(edges_to_remove)
```

```
38
39      # Step 3: Add edges with probability p_add
40      nodes = list(G.nodes())
41      for i, node1 in enumerate(nodes):
42          for node2 in nodes[i+1:]:
43              if not G.has_edge(node1, node2):
44                  if random.random() < b * (k/(n-1)):
45                      G.add_edge(node1, node2)
46
47      return G
48
49
50  def GenerateGraph(h,n,k,b):    #h = number of graphs to generate
51
52      matrices = []
53
54      for i in range(h):
55          G = ArtSimGraph(n,k,b)
56          matrix = nx.to_numpy_array(G, dtype=int)
57          matrices.append(matrix)
58
59      return np.array(matrices)
60
61
62
63  def Lambda_2(A):  #returns the second largest eigenvalue of every matrix
64      second_eigenvalues = []
65      for M in A:
66          eigenvalues = np.linalg.eigvals(M)
67          eigvals_sorted = np.sort(np.real(eigenvalues))
68          second_eigenvalues.append(eigvals_sorted[-2])
69
70
71      return second_eigenvalues
72
73
74
75  def Spectral_test(Lambda_2,n,k,c):  #returns a list of zero's (don't reject H_0) and one's (
        reject H_0).
76
77      Reject = []
78      threshold = c * max(np.sqrt(k), np.sqrt(np.log(n)))
79
80      for i in Lambda_2:
81          if i > threshold:
82              Reject.append(1)
83          else:
84              Reject.append(0)
85
86      return Reject
87
88
89  #MONTE CARLO FOR C
90  lam2_samples = Lambda_2(GenerateGraph(2000,n,k,1))
91  threshold = max(np.sqrt(k), np.sqrt(np.log(n)))
92  R = lam2_samples / threshold
93  c = np.quantile(R, 0.95)        #determining c by seeing what the ratio is for a large sample
        of ER networks.
94
95
96  #GENERATED GRAPHS & TEST STATISTICS
97  B_list = GenerateGraph(h,n,k,0)                                    #h lattice graphs.
98  A_list = GenerateGraph(h,n,k,b)                                    #h WS graphs.
99  ER_list = GenerateGraph(h,n,k,1)                                   #h ER graphs.
100 P = 1 - (1 - np.mean(np.mean(Spectral_test(Lambda_2(A_list),n,k,c)))      #percentage of
        type II errors (Power).
101 a = np.mean(Spectral_test(Lambda_2(ER_list),n,k,c))                 #(empirical)
        percentage of type I errors (alpha).
102
103
104 #PRINT
```

```
print('n =', n , 'k =', k, 'b =' ,b)
print('alpha =' , a)
print('Easy region: b =<' , max_b , 'for the spectral test to have significant power.')
print('Power =', P)
print("c_ER =", c)


#PLOTS
#1) Histogram justifying c.
plt.hist(R, bins=40, density=True, alpha=0.35, label="ER (b=1)")

for b in [0.0,0.25,0.5]: #Generating WS graphs for different b and checking the ration.
    lam2_WS = Lambda_2(GenerateGraph(2000, n, k, b))
    R_WS = lam2_WS / max(np.sqrt(k), np.sqrt(np.log(n)))
    plt.hist(R_WS, bins=40, density=True, alpha=0.35, label=f"WS b={b}")

plt.axvline(c, linestyle="--", linewidth=2, label=f"ER 95% ≈ {c:.2f}")
plt.title("Distributions of R = λ (A) / ( √k  √ log n )")
plt.xlabel("R")
plt.ylabel("Density")
plt.legend()
plt.tight_layout()
plt.show()


#2) Power and Size for different b-values.
def B_power(x,k):
    lam2_er = np.array(Lambda_2(GenerateGraph(500, n, k, 1)), dtype=float)
    thresh = max(np.sqrt(k), np.sqrt(np.log(n)))
    c = np.quantile(lam2_er / thresh, 0.95)
    return 1 - (1 - np.mean(np.mean(Spectral_test(Lambda_2(GenerateGraph(h,n,k,x)),n,k,c))))

x = np.linspace(0, 1,100)
k_values = [0.1*n, 0.5*n]

for k in k_values:
    powers = [B_power(bi,k) for bi in x]
    if k == 1*n:   # only adjust for green line
        plt.plot(x[1:-1], powers[1:-1], label=f"k = {k}")  # skip endpoints
        plt.scatter([x[0], x[-1]], [powers[0], powers[-1]], color="green")
    else:
        plt.plot(x, powers, label=f"k = {k}")


plt.axhline(0.05, color="r", linestyle="--", label="α = 0.05")
plt.xlabel("b")
plt.ylabel("Power")
plt.title(f"Power of Spectral Test vs b (n = {n})")
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.legend()
plt.show()



#3) Power and Size for different n-values.

def N_power(n,b):

    n = int(n)
    k = int(np.sqrt(n)); k += k % 2


    lam2_er = np.array(Lambda_2(GenerateGraph(500, n, k, 1)), dtype=float)
    thresh = max(np.sqrt(k), np.sqrt(np.log(n)))
    c = np.quantile(lam2_er / thresh, 0.95)

    return np.mean(Spectral_test(Lambda_2(GenerateGraph(h, n, k, b)),
                                 n, k, c))


N = np.arange(10, 101, 1)           # different graph sizes
```

```
b_values = [ 0.1, 0.25, 0.5,0.75]   # different rewiring probabilities

plt.figure(figsize=(10, 8))

for b in b_values:
    powers = [N_power(n, b) for n in N]
    plt.plot(N, powers, label=f"Power for (b={b})")

plt.axhline(a, color="r", linestyle="--", label="α = 0.05")
plt.xlabel("n")
plt.ylabel("Power")
plt.title("Power of Spectral Test for increasing n")
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.legend()
plt.show()
```

Listing A.1: Python implementation of the Spectral test.

### A.1.2. Tracy-Widom Test

```


import networkx as nx
import numpy as np
import random
import matplotlib.pyplot as plt
from TracyWidom import TracyWidom
from scipy.sparse.linalg import eigsh
from scipy.stats import kstest
from scipy.interpolate import interp1d

tw = TracyWidom(beta=1)



#FUNCTIONS GENERATING GRAPHS
def ArtSimGraph(n,k,b):  #n = number of nodes, k = tot number of neighbors, b = prob
    parameter

    G = nx.Graph()
    G.add_nodes_from(range(n))

    # Step 1: Connect each node to k/2 neighbors on each side (ring lattice)
    for node in range(n):
        for i in range(1, int((k/2)) + 1):   #If error, k could be odd.
            neighbor = (node + i) % n
            G.add_edge(node, neighbor)


      # Step 2: Remove edges with probability b
    edges_to_remove = []
    for u, v in G.edges():
        if random.random() < b:
            edges_to_remove.append((u, v))
    G.remove_edges_from(edges_to_remove)

    # Step 3: Add edges with probability p_add
    nodes = list(G.nodes())
    for i, node1 in enumerate(nodes):
        for node2 in nodes[i+1:]:
            if not G.has_edge(node1, node2):
                if random.random() < b * (k/(n-1)):
                    G.add_edge(node1, node2)

    return G


def GenerateGraph(h,n,k,b):   #h = number of graphs to generate

    matrices = []
```

```
    for i in range(h):
        G = ArtSimGraph(n,k,b)
        matrix = nx.to_numpy_array(G, dtype=int)
        matrices.append(matrix)


    return np.array(matrices)



# FUNCTIONS ESTIMATING B AND K
def even_round_bounded(x, n):
    if not np.isfinite(x):
        return 2
    k = int(np.rint(float(x)))
    k += (k % 2)
    return int(np.clip(k, 2, max(2, n - 2)))

def rrl_mask(n, k_even):
    k = int(k_even)
    if k % 2 != 0:
        raise ValueError("k_even must be even.")
    mask = np.zeros((n, n), dtype=bool)
    for s in range(1, k // 2 + 1):
        idx = np.arange(n)
        mask[idx, (idx + s) % n] = True
        mask[idx, (idx - s) % n] = True
    np.fill_diagonal(mask, False)
    return mask

def counts_LN(A, maskL):
    triu = np.triu_indices_from(A, k=1)
    L_pairs = maskL[triu]              # bool: which (i,j) are lattice-neighbor pairs
    E_pairs = A[triu].astype(bool)     # bool: which (i,j) are edges in observed A
    ML = int(np.sum(L_pairs))
    mL = int(np.sum(E_pairs & L_pairs))
    MN = int(len(L_pairs) - ML)
    mN = int(np.sum(E_pairs & (~L_pairs)))
    return mL, ML, mN, MN

def loglik_b(b, n, k, mL, ML, mN, MN, eps=1e-12):
    b = float(np.clip(b, 0.0, 1.0))
    pN = b * (k / (n - 1.0))                    # non-neighbor edge prob
    pL = 1.0 - b + (b * b) * (k / (n - 1.0))    # lattice-neighbor edge prob
    pN = np.clip(pN, eps, 1.0 - eps)
    pL = np.clip(pL, eps, 1.0 - eps)
    return (mL * np.log(pL) + (ML - mL) * np.log(1.0 - pL)
          + mN * np.log(pN) + (MN - mN) * np.log(1.0 - pN))

def mle_b_for_k(A, n, k):
    maskL = rrl_mask(n, k)
    mL, ML, mN, MN = counts_LN(A, maskL)
    # coarse grid over b, then local refine around the best point
    bs = np.linspace(0.0, 1.0, 101)
    vals = [loglik_b(b, n, k, mL, ML, mN, MN) for b in bs]
    b0 = bs[int(np.argmax(vals))]
    lo = max(0.0, b0 - 0.05); hi = min(1.0, b0 + 0.05)
    bs2 = np.linspace(lo, hi, 101)
    vals2 = [loglik_b(b, n, k, mL, ML, mN, MN) for b in bs2]
    b_hat = float(bs2[int(np.argmax(vals2))])
    ll_hat = float(np.max(vals2))
    return b_hat, ll_hat


def estimate_one_graph(A):
    n = int(A.shape[0])
    ks = list(range(2, n, 2))  # all even k

    best_ll, best_b, best_k = -np.inf, None, None
    for k in ks:
```

```
        b_hat_k, ll_k = mle_b_for_k(A, n, k)
        if ll_k > best_ll:
            best_ll, best_b, best_k = ll_k, b_hat_k, k

    return best_b, best_k, best_ll

def estimate_all_graphs(graphs, n, k_window=10):
    per_graph = []
    for A in graphs:
            b_hat, k_hat, ll = estimate_one_graph(A)  # no n, no k_window
            per_graph.append({"b": b_hat, "k": k_hat, "loglik": ll})
    b_array = np.array([d["b"] for d in per_graph], dtype=float)
    k_array = np.array([d["k"] for d in per_graph], dtype=float)
    return per_graph, b_array, k_array, float(b_array.mean()), float(k_array.mean())



#FUNCTIONS TO TEST TRACY-WIDOM DISTRIBUTION

def L_scaling(A):
    d = np.mean(A.sum(axis=1))
    return 2 + (1 / d)

def Lambda_2(A):
    # k=2 gets the largest two eigenvalues
    vals, _ = eigsh(A, k=2, which='LA' , v0=np.ones(A.shape[0]))
    return sorted(vals)[0]  # second largest

def p_hat(b,n,k):
    return b * k / (n-1)

def a_constant(p,n):
    return  np.sqrt(p / (n * (1 - p)))

def T_test(lam2,n,l,a):
    return n**(2 / 3) * (lam2 - l - a)

def Tracy_Widom(A_list,n):

    T = []

    for M in A_list:
        b, k, _ = estimate_one_graph(M)
        p0 = p_hat(1,n,k)
        a_M = a_constant(p0,n)
        lam2 = Lambda_2(M)
        L_M = L_scaling(M)
        T_M = T_test(lam2,n,L_M,a_M)
        T.append(T_M)

    return T

def t_crit(n, k, h0=500):
    # 1) simulate under H0
    G0   = GenerateGraph(h0, n, k, b=1.0)
    T0   = Tracy_Widom(G0,n)
    # 2) match to standard TW
    m, s = np.mean(T0), np.std(T0, ddof=0)
    mu  = -1.2065335745820
    var =  1.607781034581
    sigma_tw = np.sqrt(var)
    a_shift = s / sigma_tw
    b_shift = m - a_shift * mu
    # 3) set critical value
    t_TW    = tw.cdfinv(0.95)           # standard TW quantile
    t_crit  = a_shift * t_TW + b_shift      # shifted/scaled cutoff
    return t_crit

def Test(A_list, n, tcrit):
    Reject = []
```

```
    T = Tracy_Widom(A_list,n)
    for t in T:
        if t > tcrit:
            Reject.append(1) #Reject H0
        else:
            Reject.append(0) #Do not reject H0
    return Reject

def analytical_P(a0,a1,n):

       TW = TracyWidom(1)

       # 1) Calibrate affine transform under H0
       T0 = np.array(Tracy_Widom(GenerateGraph(500, n, k, b=1.0), n))
       mu, var = -1.2065335745820, 1.607781034581
       sigma_tw = np.sqrt(var)
       m0, s0   = T0.mean(), T0.std(ddof=0)
       a_shift  = s0 / sigma_tw
       b_shift  = m0 - a_shift * mu

       # 2) Critical value in observed scale
       t_alpha_std = TW.cdfinv(0.95)
       tcrit_obs   = a_shift * t_alpha_std + b_shift

       # 3) Shift term in observed scale
       delta_obs = n**(2/3) * (a0 - a1)

       # 4) Analytical power using observed distribution
       F_obs = lambda x: TW.cdf((x - b_shift)/a_shift)
       return 1 - F_obs(tcrit_obs - delta_obs)



#VARIABLES

n = 100
k = 10
h = 100


#PLOTS FOR POWER AND SIZE

#1) Power and size for different b-values

def emp_power_k(h, b, k):
    G0 = GenerateGraph(h, n, k, b)
    return np.mean(Test(G0, n, t_crit(n, k, h0=500)))

x = np.linspace(0, 1, 100)
k_values = [int(0.1 * n), int(0.5 * n)]

plt.figure(figsize=(8,6))
for k_val in k_values:
    powers = [emp_power_k(h, bi, k_val) for bi in x]
    plt.plot(x, powers, label=f"Power (k={k_val})")
plt.axhline(0.05, linestyle="--", color="red", label=f"α = 0.05)")
plt.xlabel("b")
plt.ylabel("Power")
plt.title(f"Power & Size of TW Test vs b (n = {n})")
plt.legend()
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()


#2) Power and Size for different n-values.

def emp_power_n(h, b, n):
    k_guess = max(2, int(round(0.1 * n)))
    k_even  = k_guess + (k_guess % 2)           # make even
    k_even  = min(k_even, max(2, n - 2))
    G0 = GenerateGraph(h, n, k_even, b)
```

```
    return np.mean(Test(G0, n, t_crit(n, k_even, h0=500)))

n_values = np.arange(10, 101, 1)
b_values = [0.1, 0.25, 0.5, 0.75]

for b in b_values:
    powers = [emp_power_n(h, b, n) for n in n_values]
    plt.plot(n_values, powers, label=f"Power (b={b})")

plt.axhline(0.05, linestyle="--", color="red", label=f"α = 0.05")
plt.xlabel("Number of nodes (n)")
plt.ylabel("Power")
plt.title("Power vs n for different b values (k = 0.1n)")
plt.legend()
plt.tight_layout()
plt.show()
```

Listing A.2: Python implementation of the Tracy-Widom test.

## A.2. Additional Plots

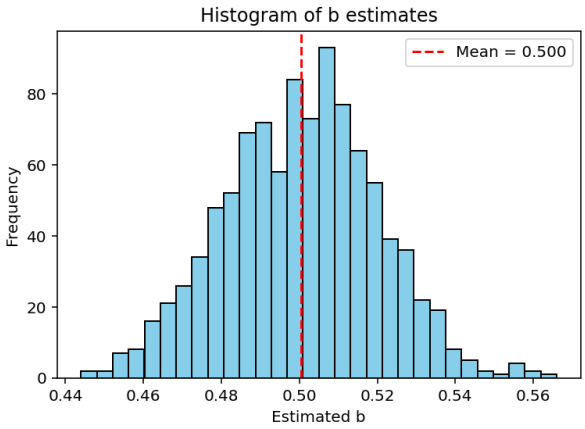


Figure A.1: Estimation for $\beta$ for the adjacency matrix with $n = 100$, $k = 10$, and $\beta = 0.5$.

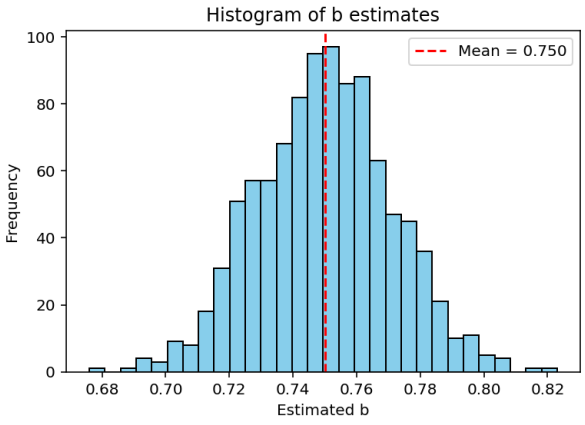


Figure A.2: Estimation for $\beta$ for the adjacency matrix with $n = 100$, $k = 10$, and $\beta = 0.75$.

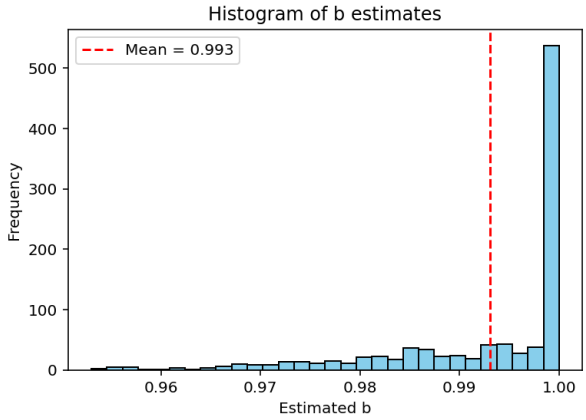


Figure A.3: Estimation for $\beta$ for the adjacency matrix with $n = 100$, $k = 10$, and $\beta = 1$ (ER graph).

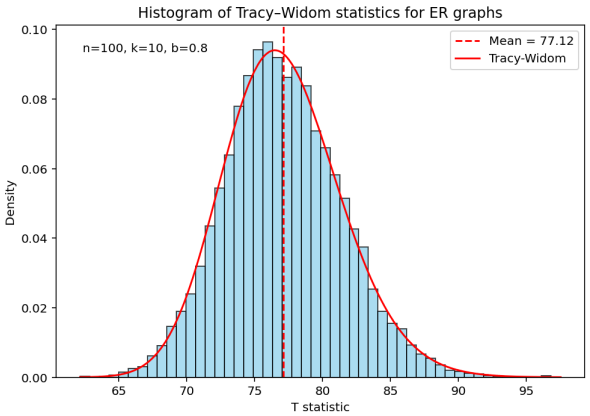


Figure A.4: Histogram of $T$ for $\beta = 0.8$, $n = 100$, and $k = 10$. The red line is the scaled TW distribution.

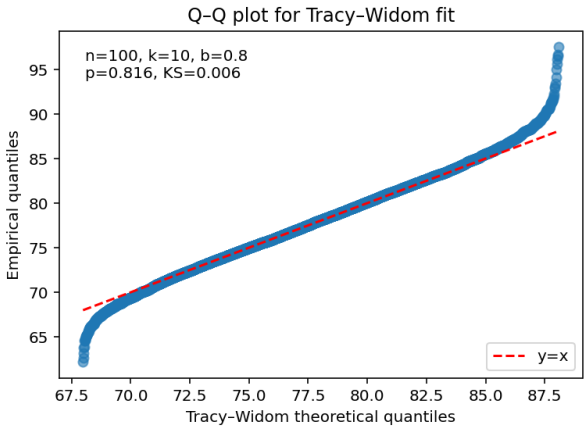


Figure A.5: QQ-plot to visualize the overlap between the TW distribution and $T$ for $\beta = 0.8$, $n = 100$, and $k = 10$. The $p$- and KS-values are reported.

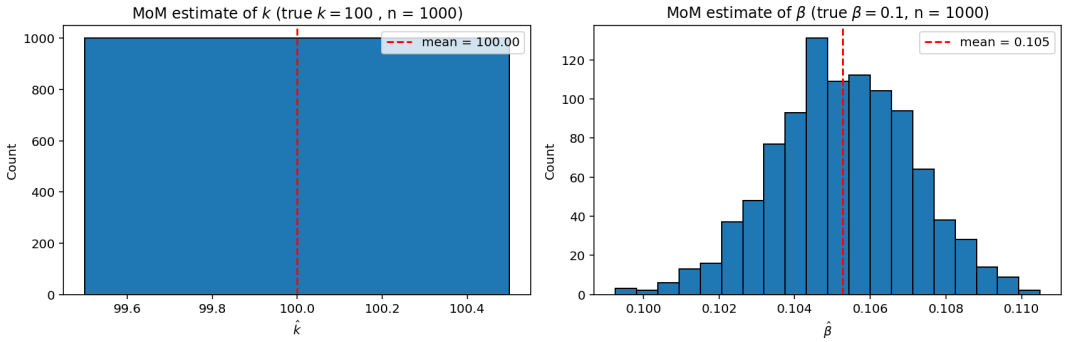


Figure A.6: Histogram of $k$ and $\beta$ estimates using the MoM. 1000 graphs are generated, each with $n = 100$ and true value $k = 100$ and $\beta = 0.1$.

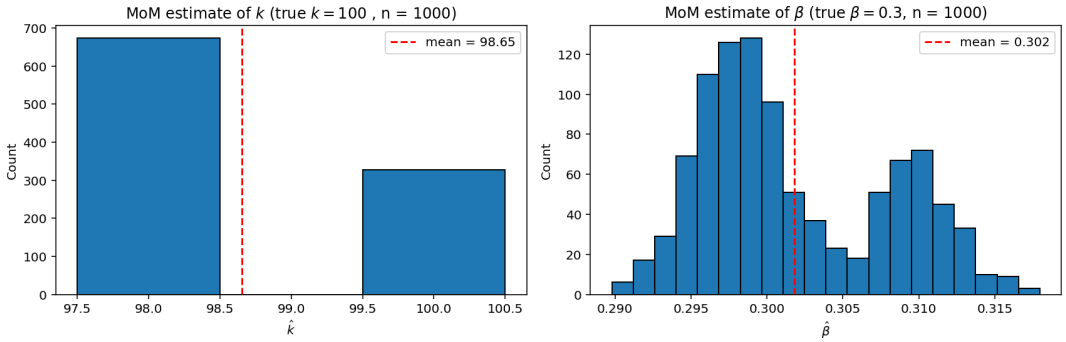


Figure A.7: Histogram of $k$ and $\beta$ estimates using the MoM. 1000 graphs are generated, each with $n = 100$ and true value $k = 100$ and $\beta = 0.3$.

# Bibliography


Amaral, L. A. N., Scala, A., Barthélemy, M., and Stanley, H. E. (2000). Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152.

Bijma, F., Jonker, M., and Van der Vaart, A. (2026). *An Introduction to Mathematical Statistics*. Amsterdam University Press, Amsterdam.

Bornemann, F. (2010). On the numerical evaluation of distributions in random matrix theory: A review. *Markov Processes and Related Fields*, 16(4):803–866.

Cai, T., Liang, T., and Rakhlin, A. (2017). On detection and structural reconstruction of small-world random networks. *IEEE Transactions on Network Science and Engineering*, 4(3):165–176.

Chauhan, S., Girvan, M., and Ott, E. (2009). Spectral properties of networks with community structure. *Physical Review E*, 80(5):056114.

Cugliandolo, L. F. (2019). Lecture notes on the gaussian orthogonal ensemble (goe). https://www.lpthe.jussieu.fr/~leticia/TEACHING/Master2019/GOE-cuentas.pdf. Accessed: October 1, 2025.

Erdős, L., Knowles, A., Yau, H.-T., and Yin, J. (2013). Spectral statistics of erdős–rényi graphs i: Local semicircle law. *The Annals of Probability*, 41(3B):2279–2375.

Fabris, M. (2023). On the characterization of regular ring lattices and their relation with the dirichlet kernel. *arXiv preprint arXiv:2304.11602*.

Hodson, T. O. (2022). Root-mean-square error (rmse) or mean absolute error (mae): when to use them or not. *Geoscientific Model Development*, 15(14):5481–5487.

Lee, J. O. and Schnelli, K. (2016). Local law and tracy–widom limit for sparse random matrices. *arXiv preprint arXiv:1605.08767*.

Lehmann, E. L. and Romano, J. P. (2005). *Testing Statistical Hypotheses*. Springer, New York, NY, 3rd edition.

Marcozzi, A. (2017). Lecture 13: Small-world and scale-free networks. https://cseweb.ucsd.edu/classes/wi17/cse158-a/slides/lecture13_annotated.pdf. Accessed: 20 November 2025.

Meghanathan, N. (2015). Small-world networks. https://www.jsums.edu/nmeghanathan/files/2015/08/CSC641-Fall2015-Module-6-Small-World-Networks-reduced.pdf. Lecture slides, Jackson State University.

Morris, N. J. and Elston, R. C. (2012). A note on comparing the power of test statistics at low significance levels. *The American Statistician*, 66(3):130–134.

Motzkin, T. S. and Taussky, O. (1955). Pairs of matrices with property l. ii. *Transactions of the American Mathematical Society*, 80(2):387–401.

Newman, M. E. J. (2002). Spread of epidemic disease on networks. *Physical Review E*, 66(1):016128.

Nordgren, R. P. (2020). Simultaneous diagonalization and svd of commuting matrices. *arXiv preprint arXiv:2006.16364*.

Pastor-Satorras, R. and Vespignani, A. (2001). Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203.

Slutsky, D. J. (2014). The effective use of graphs. *Journal of Wrist Surgery*, 3(2):67–68.

Stanley Milgram (2025). Stanley milgram. https://psychology.fas.harvard.edu/people/stanley-milgram. Harvard University Department of Psychology. Accessed: 17 September 2025.

Strang, G. (2017). Circulant matrices (mit 18.06 linear algebra lecture notes). https://web.mit.edu/18.06/www/Spring17/Circulant-Matrices.pdf.

Tracy, C. A. and Widom, H. (1994). Level-spacing distributions and the airy kernel. *Communications in Mathematical Physics*, 159(1):151–174.

Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, 32(4):425–443.

Tsourakakis, C. E. (2008). Fast counting of triangles in large real networks without counting: Algorithms and laws. *Theoretical Computer Science*, 407(1–3):318–326.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Weisstein, E. W. (2025). Scale-free network. https://mathworld.wolfram.com/Scale-FreeNetwork.html. From MathWorld–A Wolfram Web Resource.

Wigner, E. P. (1958). On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, 67(2):325–327.

Zeimbekakis, A. (2022). n misuses of the k on misuses of the kolmogor olmogorov–smirno v–smirnov test for one-sample est for one-sample goodness-of-fit. https://digitalcommons.lib.uconn.edu/cgi/viewcontent.cgi?article=1900&context=srhonors_theses. Accessed: 20 November 2025.