University of Hradec Králové

Faculty of Science

Department of Physics

Quantum chaos on graphs

Bachelor's thesis

Author: ŠTĚPÁN ZELENKA
Field of study: Fyzika se zaměřením na vzdělávání

Supervisor: doc. RNDr. JIŘÍ LIPOVSKÝ, Ph.D.

Hradec Králové 2023

Univerzita Hradec Králové
Přírodovědecká fakulta

Zadání bakalářské práce

Autor:	**Štěpán Zelenka**
Studium:	S20FY007BP
Studijní program:	B0114A110004 Fyzika se zaměřením na vzdělávání
Studijní obor:	Informatika se zaměřením na vzdělávání, Fyzika se zaměřením na vzdělávání
Název bakalářské práce:	**Kvantový chaos na grafech**
Název bakalářské práce AJ:	Quantum chaos on graphs

Cíl, metody, literatura, předpoklady:

Chaotické chování bývá v kvantové teorii určováno pomocí statistiky rozložení vlastních čísel problému (konkrétně statistiky vzdáleností dvou nejbližších vlastních čísel). Bylo ukázáno, že pro některé kvantové grafy (například pro úplné grafy) přibližuje se statistika jejich vlastních hodnot numericky statistice vlastních hodnot některých typů náhodných matic.

Úkolem studenta bude ověřit dříve získané poznatky o statistickém chování vlastních hodnot kvantových grafů. Nejdříve nalezne statistiku vlastních hodnot tří základních skupin náhodných matic: gaussovských ortogonálních, gaussovských unitárních a gaussovských symplektických matic. Poté nalezne statistiku pro vybrané kvantové grafy a porovná ji s tou získanou v teorii náhodných matic. Téma je vhodné pro studenta s kladným vztahem k programování. Více podrobností např. v [1], [2], [3].

[1] G. Berkolaiko, P. Kuchment, Introduction to Quantum Graphs, Mathematical Surveys and Monographs 186; 270 pp; hardcover. AMS (2013).

[2] M. L. Mehta, Random Matrices, vol. 142 of Pure and Applied Mathematics, Academic Press, 3rd ed. (2004).

[3] G. Berkolaiko, E. B. Bogomolny, and J. P. Keating, Star graphs and Šeba billiards, J. Phys. A, 34 (3): 335-350 (2001), arXiv: 0010045 [nlin.CD], https://arxiv.org/abs/nlin/0010045

Zadávající pracoviště:	Katedra fyziky, Přírodovědecká fakulta
Vedoucí práce:	doc. RNDr. Jiří Lipovský, Ph.D.
Oponent:	doc. RNDr. Jan Kříž, Ph.D.

Datum zadání závěrečné práce: 11.8.2021

Declaration

I declare that I prepared the bachelor thesis independently and with using the mentioned literature.

. . . . . .

Štěpán Zelenka

July 25, 2023

Acknowledgements

I am grateful to my supervisor, doc. RNDr. Jiří Lipovský, PhD., for his valuable advice and rigorous supervision.

Annotation

ZELENKA, Štěpán. *Quantum chaos on graphs.* Hradec Králové: University of Hradec Kralove, Faculty of Science, Department of Physics, 2023. 52 p.

The thesis deals with the issue of quantum graphs, exploring their properties and chaotic behavior. The reader is first introduced to various branches and methods of investigating quantum chaos. Following that are chapters dedicated to the theory of random matrices and quantum graphs, describing the models utilized in the sixth chapter. A chapter of the thesis is also devoted to the problem of finding function roots. The final chapter examines the statistical properties of selected random matrices and quantum graphs.

Keywords

Quantum graphs, quantum chaos, random matrix theory, nearest neighbor distribution

Anotace

ZELENKA, Štěpán. *Kvantový chaos na grafech.* Hradec Králové: Univerzita Hradec Králové, Přírodovědecká fakulta, Katedra fyzika, 2023. 52 s.

Práce pojednává o problematice kvantových grafů, zabývá se jejich vlastnostmi a chaotickým chováním. Nejdříve je čtenář seznámem s různými odvětvími a metodami zkoumání kvantového chaosu. Následují kapitoly věnované teorii náhodných matic a kvantovým grafům, popisující modely využívaný v šesté kapitole. Část práce je také věnovaná problematice hledání kořenů funkce. V poslední kapitole jsou zkoumány statistické vlastnosti vybraných náhodných matic a kvantových grafů.

Klíčová slova

Kvantové grafy, kvantový chaos, teorie náhodných matrix, rozdělení nejbližších sousedů

Contents

Introduction

The study of quantum chaos has emerged as a captivating field, unraveling the intricate interplay between classical chaos and quantum mechanics. In this thesis, we will use quantum graphs as a model to study quantum graphs, as proposed by Kottos and Smilansky in [1]. This methodology is based on the Bohigas-Giannoni-Schmit (BGS) conjecture [2], which states that some statistical properties of a chaotic spectra can be predicted by certain random matrices. This conjecture is supported by a vast host of numerical studies.

The goal of this thesis is to give the reader a brief introduction to the issue of quantum chaos on graphs and the study of their properties. Specifically, the nearest neighbor distribution (NND) will be studied for selected random matrices and quantum graphs, illustrating their relationship, which will then serve as a numerical proof of the BGS conjecture. The thesis will also provide an exploration of random matrices, with a particular emphasis on the Gaussian Ensemble. This investigation aims to deepen the understanding of the BGS conjecture.

Chapter 1

Quantum chaos

1.1 Classical chaos

In physics and mathematics, chaos theory describes the behavior of nonlinear dynamic systems, meaning systems that evolve over time and the response of the system is not proportional to the magnitude of the input. Under certain conditions, these systems may exhibit an effect referred to as deterministic chaos. One of its main characteristic is its sensitivity to initial conditions of the system. As a result of this sensitivity, the behavior of these systems may appear chaotic and random, despite the fact that the model of the system is deterministic in the sense that it is completely defined and does not include any random variables. An example of such system is a double pendulum; the position of the pendulum after a certain amount of time may be completely different for a set of nearly identical initial conditions, displaying the chaotic, yet deterministic (can be described by a set of coupled ordinary differential equations), behavior [3].

Understanding chaotic behavior in physical systems is of great importance in several fields of science and engineering. In physics, chaotic behavior is a ubiquitous phenomenon that can be found in a wide range of systems, from classical mechanics to quantum mechanics. Chaotic behavior can occur in systems as diverse as the weather, the motion of planets, the behavior of fluids, and the dynamics of electronic circuits. By understanding the chaotic behavior of these systems, scientists and engineers can gain insights into their fundamental properties and develop more accurate models to predict their behavior.

In addition, understanding chaotic behavior is important for the design and control of complex systems. Chaotic behavior can lead to unexpected and unpredictable outcomes in engineering systems such as aircraft, bridges, and power grids. Therefore, engineers must be able to predict and control the behavior of these systems to ensure their safety and reliability.

Moreover, the study of chaotic behavior has led to the development of new mathematical tools and concepts that have found applications in fields such as cryptography, data compression, and image processing. The importance of understanding chaotic behavior in physical systems is thus far-reaching and has implications for many areas of science and engineering.

1.2 Quantum chaos

When we begin to study chaotic systems in terms of quantum mechanics, we start seeing certain differences. According to the Heisenberg's uncertainty principle, it is impossible to measure certain pairs of physical quantities of a particle, such as position and momentum, with arbitrary precision.[4] This means that quantum systems are not deterministic, in contrast to classical systems. In quantum mechanics, the state of a system is given by the wave function and the Schrödinger equation describes its evolution over time. The Schrödinger equation is a linear partial differential equation, meaning that the evolution of a quantum system is governed by a linear equation, which results in a linear transformation of the wave function describing the system. Therefore, the evolution of quantum systems is fundamentally different from that of classical systems, where nonlinearity can lead to chaotic behavior. However, when the system has a large number of degrees of freedom or when it is subject to external perturbations, it may exhibit chaotic behavior.[5]

The relationship between classical and quantum chaos is in part described by the correspondence principle. It is a fundamental concept in physics that states that in the limit of large quantum numbers, the behavior of quantum systems should be consistent with classical mechanics. This principle implies that classical mechanics is a special case of quantum mechanics and that classical behavior can emerge from quantum behavior.

In the context of chaos theory, the correspondence principle suggests that classical chaotic behavior should have a quantum mechanical counterpart. This is known as quantum chaos and the primary question that it seeks to answer is: *What is the relationship between quantum mechanics and quantum chaos?* However, the relationship between classical and quantum chaos is not straightforward. While classical chaotic behavior arises from nonlinearity, quantum mechanics is a linear theory. Therefore, quantum chaos cannot be fully understood as a simple extension of classical chaos. Instead, the study of quantum chaos involves finding ways in which classical and quantum mechanics are related. [6]

Despite the challenges, the study of quantum chaos is an important area of research with numerous applications. For example, quantum chaos has been used to study the behavior of complex systems in condensed matter physics, as well as to develop new quantum computing algorithms. It has also shed light on the nature of the correspondence principle and the relationship between classical and quantum mechanics.

1.3 Methods

To bring answers to the basic question of quantum chaos, several approaches have been employed.

1.3.1 Semiclassical quantization

Semiclassical quantization is a powerful tool in the study of quantum chaos, and has been applied to a wide range of physical systems, from atoms and molecules to black holes and cosmology.

The semiclassical approximation is based on the idea that, in the limit of large quantum numbers, the quantum dynamics of a system approach the classical dynamics. This means that we can use classical mechanics to obtain an approximate expression for the energy levels of a quantum system, by expanding the wave function in powers of Planck's constant and then using classical mechanics to determine the coefficients of the expansion.

The method was first developed by Michael Berry in the late 1970s, and has since been widely applied in the study of quantum chaos. One of the key applications of semiclassical quantization is in the study of the spectral statistics of chaotic systems. The spectral statistics describe the distribution of energy levels of a quantum system, and can reveal important information about the underlying classical dynamics.

The semiclassical approximation has also been used to study other properties of chaotic systems, such as quantum transport, quantum localization, and quantum chaos in cosmology. In recent years, the method has been combined with other techniques, such as Random Matrix Theory (RMT) and the theory of wave chaos, to provide even deeper insights into the behavior of complex quantum systems. [7]

1.3.2 Quantum scars

The study of scars and periodic orbits is a powerful tool for understanding the relationship between classical and quantum mechanics in chaotic systems. Periodic orbits are closed trajectories in the phase space of a classical system that repeat themselves after a certain period of time. In a chaotic system, periodic orbits can be unstable, meaning that small perturbations can cause the trajectory to diverge from the original path. The concept of scars refers to a peculiar phenomenon where certain quantum states are concentrated around periodic orbits that are classically unstable.

The study of scars and periodic orbits was initiated by Eric Heller and his colleagues in the early 1990s. They found that the wave functions of certain quantum systems exhibit striking patterns that are correlated with the unstable periodic orbits of the corresponding classical systems. These patterns, which they dubbed scars, represent a deviation from the random distribution of wave functions that is characteristic of chaotic systems.

Scars arise due to the interference of wave functions that are associated with different periodic orbits. When the phase differences between these wave functions are carefully chosen, they can add up constructively to produce a coherent, localized state that is centered around the unstable periodic orbit. The resulting scar is a manifestation of the classical motion of the system in the quantum realm.

The distribution of scars in a quantum system can provide valuable information about the underlying classical dynamics. For example, the number and location of scars can reveal the presence of periodic orbits and the degree of chaos in the system. In particular, systems with weak chaos tend to have more isolated scars, while systems with strong chaos have scars that are more widely distributed.

The study of scars and periodic orbits has led to numerous advances in our understanding of the interplay between classical and quantum mechanics in chaotic systems. It has provided insight into the fundamental limits of quantum mechanics, the nature of decoherence, and the emergence of classical behavior from quantum systems. Moreover, the identification of scars has important applications in a wide range of fields, including quantum computing, mesoscopic physics, and chemical dynamics. [8]

1.3.3 Entanglement entropy

Entanglement entropy is a method used in the study of quantum chaos that involves analyzing the amount of entanglement between different parts of a quantum system. Entanglement is a quantum mechanical phenomenon in which the properties of two or more particles become intertwined, even when they are separated by a large distance. This means that the

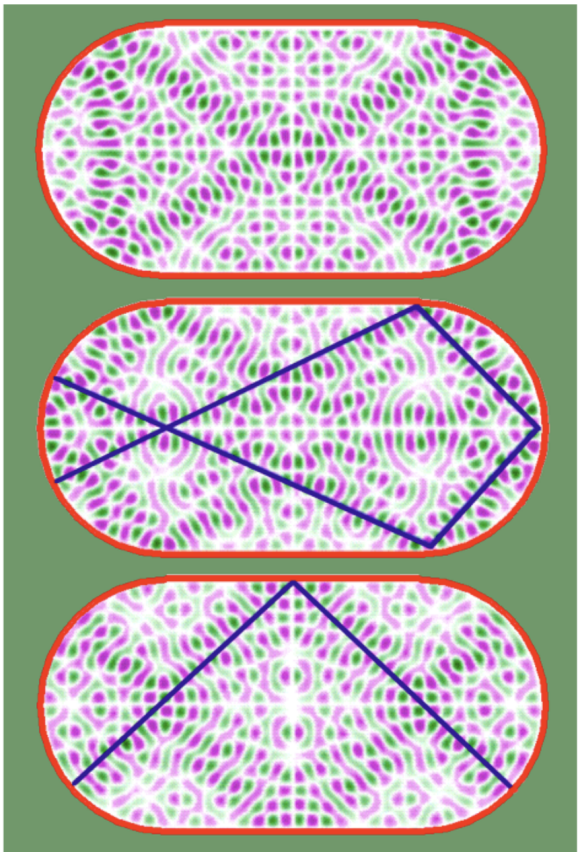

Figure 1.1: Typical scarred eigenstates of the (Bunimovich) stadium. The figure shows the probability density for three different eigenstates. The scars, referring the regions of concentrated probability density, are generated by (unstable) periodic orbits, two of which are illustrated.[9]

state of one particle cannot be described independently of the other, and the properties of the particles are said to be "entangled".

In the context of quantum chaos, entanglement entropy is a measure of the amount of entanglement between different parts of a quantum system. It is defined as the von Neumann entropy of the reduced density matrix of a subsystem of the full system. The von Neumann entropy is a measure of the amount of information contained in a quantum state, and it is related to the amount of uncertainty that exists about the state of the system.

In chaotic quantum systems, the entanglement entropy grows rapidly as the size of the subsystem increases. This is due to the fact that chaotic systems are highly entangled, with the entanglement between different parts of the system spreading rapidly throughout the entire system. This rapid growth of entanglement entropy is a signature of chaotic behavior in quantum systems.

The study of entanglement entropy in quantum chaotic systems has led to a better understanding of the relationship between chaos and entanglement in quantum mechanics. It has also been used to investigate the properties of black holes and their relationship to quantum chaos.

One of the key insights gained from the study of entanglement entropy is the concept of "quantum chaos versus thermalization". Thermalization is the process by which a system

reaches thermal equilibrium with its environment. In quantum mechanics, it is related to the concept of decoherence, which is the loss of coherence between different parts of a quantum system due to its interaction with the environment.

In chaotic quantum systems, it was initially thought that the rapid growth of entanglement entropy would lead to thermalization, as the system becomes more and more entangled with its environment. However, recent studies have shown that this is not always the case, and that some quantum systems can exhibit "quantum chaos" without reaching thermal equilibrium. This has important implications for the study of quantum many-body systems and the behavior of black holes. [10]

Overall, the study of entanglement entropy in quantum chaotic systems has provided a powerful tool for understanding the complex behavior of these systems. It has led to new insights into the relationship between chaos and entanglement in quantum mechanics, and has opened up new avenues of research in areas such as black hole physics and quantum information theory.

1.3.4 Spreading of wave packets

Another characteristic of quantum systems is the spreading of wave packets. In a quantum system, the motion of a particle is described by a wave function. When the wave function is spread out, it indicates that the particle is delocalized and has a high probability of being found in many different locations. Conversely, when the wave function is localized, the particle is more likely to be found in a specific location. In chaotic quantum systems, the spreading of wave packets is rapid and follows a specific pattern. This is known as quantum revivals, where the wave function spreads out and then recoheres periodically. The time scale of the revivals is related to the classical dynamics of the system. In regular quantum systems, the wave function spreads out more slowly and does not exhibit quantum revivals.

By analyzing the spreading of wave packets in a quantum system, we can identify whether the corresponding classical system exhibits chaotic or regular behavior. This method has been used to study a wide range of physical systems, including billiards, quantum dots, and atoms in strong electromagnetic fields. [11]

1.3.5 Spectral statistics

Eigenvalue statistics in quantum systems have been studied extensively in the field of quantum chaos. One of the key observations in this area is that the statistical behavior of eigenvalues can be used to determine the chaotic behavior of the system. This is because the eigenvalues of a quantum system are related to the energy levels of the system, and the energy levels of a chaotic system are known to exhibit a specific statistical behavior.

The study of eigenvalue statistics in quantum systems can be traced back to the work of Eugene Wigner in the 1950s. Wigner was interested in the statistical behavior of the eigenvalues of large matrices that arise in the study of nuclear physics. He observed that the eigenvalues of these matrices exhibit a semicircular distribution in the limit of large matrix size. This result, known as Wigner's semicircle law, is now considered one of the fundamental results in RMT.

In quantum chaos, the focus is on the statistical behavior of the eigenvalues of Hamiltonian that describe the evolution of quantum systems over time. In chaotic systems, the eigenvalues of the Hamiltonian exhibit a statistical behavior that is different from that of non-chaotic systems. Specifically, the eigenvalues of chaotic systems exhibit a distribution that is similar to that of the eigenvalues of random matrices.

The statistical behavior of eigenvalues in quantum systems has many important applications. For example, it can be used to understand the spectral properties of disordered systems, such as amorphous materials or glasses. It can also be used to understand the behavior of quantum systems in the presence of external perturbations or noise. In addition, the statistical behavior of eigenvalues has important applications in quantum computing and quantum information theory. [12]

In later chapters, the focus will be on this method, specifically the nearest-neighbor level spacing distribution of random matrices and quantum systems.

Chapter 2

Random matrix theory

In RMT, a random matrix is a matrix in which some or all values are random variables. Random matrices become useful when we want to statistically describe a complex physical or mathematical system, where we replace the deterministic matrices with random matrices and then calculate averages and other statistical properties.

2.1 Gaussian Ensemble

The Gaussian Ensemble is a class of random matrices that plays an important role in RMT and its applications to physics, mathematics, and other fields. The ensemble consists of matrices whose elements are independent and identically distributed (i.i.d.) Gaussian random variables, with zero mean and variance depending on the specific symmetry class of the ensemble.

The Gaussian Ensembles are characterized by the probability density function (PDF) of the matrix elements. The three most commonly studied ensembles are the real symmetric Gaussian Orthogonal Ensemble (GOE), the complex Hermitian Gaussian Unitary Ensemble (GUE), and the quaternion self-dual Gaussian Symplectic Ensemble (GSE). In GOE, the matrix elements are real and symmetric. Its distribution is invariant under orthogonal conjugation, meaning

$$H \to W^T H W$$

where W is any real orthogonal matrix of order N. The PDF for the matrix elements is given by:

$$P_{\mathrm{GOE}}(X) \propto \exp\left(-\frac{N}{4}\mathrm{Tr}X^2\right),$$

where N is the dimension of the matrix, Tr denotes the trace of the matrix, i.e., the sum of its diagonal elements, and X is an $N \times N$ matrix.

In GUE, the matrix elements are complex and Hermitian (the matrix is equal to its conjugate transpose). Its distribution is invariant under unitary conjugation, meaning

$$H \to U^{-1} H U.$$

The PDF for the matrix elements is given by:

$$P_{\mathrm{GUE}}(X) \propto \exp\left(-\frac{N}{2}\mathrm{Tr}X^2\right).$$

In GSE, the matrix elements are quaternion self-dual, meaning that the matrix is equal to its conjugate transpose up to a sign. Its distribution is invariant under conjugation by the symplectic group, meaning

$$H \to W^T H W,$$

where W is any symplectic matrix of order N and W^T is its transpose. A symplectic matrix is a $2n \times 2n$ matrix with real entries that satisfies the condition

$$W^T \Omega W = \Omega,$$

where Ω is a fixed $2n \times 2n$ invertible, skew-symmetric matrix. The PDF for the matrix elements is given by:

$$P_{\text{GSE}}(X) \propto \exp\left(-\frac{N}{4}\text{Tr}X^2 + \frac{N}{8}\text{Tr}X^4\right).$$

[13]

Note that the Gaussian Ensembles are denoted by their Dyson index, $\beta = 1$ for GOE, $\beta = 2$ for GUE, and $\beta = 4$ for GSE. This index counts the number of real components per matrix element.

2.2 Wigner's semicircle law

Wigner's semicircle law is a fundamental result in RMT that describes the distribution of eigenvalues in large random matrices. Specifically, it states that the normalized eigenvalue density of a large, symmetric or Hermitian matrix with independent, identically distributed entries will converge to a semicircle distribution as the size of the matrix goes to infinity.

More precisely, suppose we have an $N \times N$ real symmetric matrix A whose entries above the diagonal are independent random variables with mean zero and variance $1/N$, and whose diagonal entries are independent random variables with mean zero and variance $2/N$. Then, as N becomes large, the density of the eigenvalues of A converges to the semicircle density

$$\frac{2}{\pi R^2}\sqrt{R^2 - x^2}, \text{if } -R \leq x \leq R,$$

where $R = \sqrt{2N}$ is the radius of the semicircle. In other words, the probability that an eigenvalue of A falls in the interval $[a, b]$ is given by the area under the semicircle between a and b, divided by the total area of the semicircle. Then suppose we are interested in the distribution of a single arbitrary eigenvalue E_1 of a Gaussian Ensemble with Dyson index β as $N \to \infty$, then Wigner's semicircle is

$$\lim_{N\to\infty} \sqrt{\beta N}\rho(\sqrt{\beta N}E_1) = \sigma_{\sqrt{2}}(E_1), \tag{2.1}$$

where ρ is the PDF of the distribution and σ is the standard deviation of the Wigner semicircle distribution. The semicircle law holds for a wide variety of ensembles of random matrices, including the Gaussian orthogonal, unitary, and symplectic ensembles. [14]

2.3 Wigner's surmise

Wigner's surmise is a key result in RMT that describes the distribution of spacings between adjacent eigenvalues of certain random matrices. In particular, the surmise provides a

formula for the PDF of the NND of ensembles of random matrices that exhibit certain types of symmetries, such as orthogonal, unitary, or symplectic symmetry. The surmise was first proposed by Eugene Wigner in 1951, based on numerical evidence from studies of the spectra of heavy nuclei. It states that the NND can be approximated by a function that has a characteristic shape, depending on the symmetry class of the ensemble. The surmise for the NND of GOE is given by a semi-circular function [12]

$$P_1(s) = \frac{\pi}{2} s \exp\left(-\frac{\pi}{4} s^2\right). \tag{2.2}$$

The surmise for the GUE is given by a function that is quadratic at small spacings and has a long tail at large spacings. [14]

$$P_2(s) = \frac{32}{\pi^2} s^2 \exp\left(-\frac{4}{\pi} s^2\right). \tag{2.3}$$

For GSE, the PDF approaches a modified Wigner surmise, known as the Dyson surmise [15]

$$P_4(s) = \frac{2^{18}}{3^6 \pi^3} s^4 \exp\left(-\frac{64}{9\pi} s^2\right). \tag{2.4}$$

Here, s is the normalized spacing between adjacent eigenvalues, i.e. $s = S/D$, where S is the difference between two neighbouring eigenvalues and D is the average spacing. The Wigner surmise and Dyson surmise are normalized such that the integral over all possible spacings equals one. The surmise has since been verified in many other contexts, both theoretically and experimentally, and is now widely used as a tool for analyzing the spectral properties of complex systems in physics, mathematics, and other fields.

2.4 Unfolding

When studying the spacings of eigenvalues of a random matrix, we have to take into account the fact that the eigenvalues are not distributed uniformly, but rather according to Wigner's semicircle law, as described above. So in order to study the behavior of spacings asymptotically, we need to unfold the eigenvalues. The unfolded eigenvalues will then be distributed uniformly, with mean spacing one. Only after this operation, the NND will follow the Wigner's surmise.

Suppose we have an order sequence of energies (eigenvalues) which form the spectral function

$$S(E) = \sum_{n=1}^{N} \sigma(E - E_n),$$

where σ is the Dirac delta function. To perform the unfolding, we will use the average level staircase function

$$\eta(E) = \int_{E}^{-\infty} S(E')dE',$$

where E is an eigenvalue. This tells us how many eigenvalues of all the eigenvalues are less than E. It is then decomposed into a smooth part $\xi(E)$ and a fluctuating part $\eta_{\mathbb{fl}}(E)$,

$$\eta(E) = \xi(E) + \eta_{\mathbb{fl}}(E).$$

The smooth part is given by the cumulative mean level density,

$$\xi(E) = \int_E^{-\infty} R_1(E')dE',$$

where R_1 is the mean level density, which refers to the average density of energy levels or eigenvalues of a quantum system over a certain range. The unfolded sequence of eigenvalues is then given by

$$\xi_i = \xi(E_i),$$

where the index i labels the eigenvalues in the sequence. [16]

Now, to find the scaling factor for the unfolding, we will use Wigner's semicircle law. Note that the equation (2.1) implies that $\frac{E_1}{\sqrt{\beta N}}$ tends to a semicircle distribution with radius $\sqrt{2}$ as N gets large. Or in other form, E_1 tends to a distribution with radius $\sqrt{2\beta N}$. Let $\Lambda_{N,\beta}[a;b]$ be the number of eigenvalues in the interval $[a;b]$ of an $N \times N$ Gaussian Ensemble with Dyson index β, then

$$\lim_{N\to\infty} \Lambda_{N,\beta}[a;b] \xrightarrow{p} N \int_a^b \sigma_{\sqrt{2\beta N}}(t)dt,$$

where $\xrightarrow{p}$ denotes convergence in probability. This means that as N approaches infinity, the probability that $\Lambda_{N,\beta}[a;b]$ deviates from the expected value $N\int_a^b \sigma_{\sqrt{2\beta N}}(t)dt$ goes to zero. We can then find a closed form expression for the integral

$$\rho_{\sqrt{2\beta N}}(t) = \frac{N\beta \arcsin(\frac{\sqrt{2}t}{2\sqrt{N\beta}}) + \frac{t\sqrt{2N\beta - t^2}}{2}}{\pi\beta}.$$

Now we define the mean cumulative spectral density

$$\xi_{N,\beta}(E) = N \int_{-2\beta N}^{E} \rho_{\sqrt{2\beta N}}(t)dt$$

and the closed form expression is

$$\xi_{N,\beta}(E) = \frac{N}{2} + \frac{N\beta \arcsin \frac{\sqrt{2}E}{2\sqrt{N}\sqrt{\beta}} + \frac{E\sqrt{2N\beta - E^2}}{2}}{\pi\beta}.$$

Thus we get

$$\xi_{N,\beta}(E) = \begin{cases} 0 & \text{for } E \le \sqrt{-2\beta N} \\ \frac{N}{2} + \frac{1}{\pi\beta}(N\beta \arcsin(\frac{\sqrt{2}E}{2\sqrt{N\beta}}) + \frac{E\sqrt{2\beta N - E^2}}{2}) & \text{for } E \in (\sqrt{-2\beta N}, \sqrt{2\beta N}) \\ 1 & \text{for } E \ge \sqrt{2\beta N}. \end{cases} \quad (2.5)$$

We can now get the unfolded sequence

$$\xi_i = \xi_{N,\beta}(E_i),$$

which will have a mean equal to one. [17]

Chapter 3

Quantum graphs

Quantum graph is a metric graph equipped with a differential operator, which we call Hamiltonian, and a set of some vertex conditions (which are described below). Quantum graphs arise naturally as simplified models in various areas of mathematics, physics, chemistry and engineering, when one studies propagation of waves of various nature through quasi-one-dimensional system, for example quantum wires, photonic crystals and the free-electron theory of conjugated molecules. Quantum graphs also play a role of simplified models for studying quantum chaos.

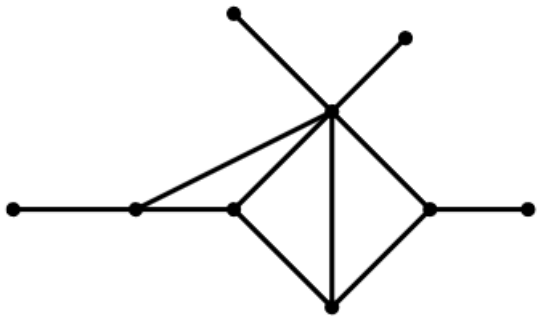

Figure 3.1: A graph.

3.1 Describing the model

A graph Γ consists of a set of vertices $\mathcal{V} = \{\nu_i\}$ and a set of edges $\mathcal{E} = \{e_j\}$ connecting the vertices. The edges are undirected. We will use the notation $E := |\mathcal{E}|$ and $V := |\mathcal{V}|$ for the number of edges and vertices. The edges are of a finite length $l_j > 0$ and can be parameterized by the intervals $(0, l_j)$. This metric graph becomes a quantum one after being equipped with a differential operator and some vertex conditions. This operator is called the Hamiltonian $\mathcal{H}$ which acts as a negative second derivative on each edge

$$f(x) \to -\frac{d^2 f}{dx^2}, \tag{3.1}$$

where x is the coordinate x along an edge. For the definition of an operator to be complete, we need to describe its domain. The domain of operator $\mathcal{H}$ are functions from the Sobolev space $H^2(e)$ on each edge e. These are functions for which the first and the second derivative exist and both the components of these function on each edge and their first and second derivatives are quadratically integrable, meaning

$$\int_0^{l_j} |f_j(x)|^2 dx, \ \int_0^{l_j} |f_j'(x)|^2 dx \text{ and } \int_0^{l_j} |f_j''(x)|^2 dx$$

are finite. The next condition is that the function satisfies the vertex conditions. Some of the most common vertex conditions are: [18]

- Dirichlet condition – we impose a condition on all edges in a given vertex ν

  $$f_j(\nu) = 0.$$

  This means that in this vertex, the graph is disconnected.

- Neumann condition – we impose a condition on all edges in a given vertex ν

  $$f_j'(\nu) = 0.$$

  This also means that the graph is disconnected.

- δ-condition – function in a given vertex satisfies the conditions

  $$f_i(\nu) = f_j(\nu) = f(\nu),$$

  where $i, j \in \{1, ..., d(\nu)\}$, $d(\nu)$ being the degree of a vertex ν.

  $$\sum_{j=1}^{d(\nu)} f_j'(\nu) = \alpha f(\nu).$$

  This means that functions in the given vertex are continuous and the sum of outgoing derivatives is equal to α-multiple of the function value, constant α is a real number.

- Standard condition – the condition is the same as the δ condition (functions in a given vertex are continuous), but the value of α is zero, meaning the sum of outgoing derivatives is zero.

  $$\sum_{j=1}^{d(\nu)} f_j'(\nu) = 0.$$

3.2 Finding the secular determinant

If we want to study the NND of eigenvalues, we first need to derive the equation for the eigenvalues (the energy levels). In quantum mechanics, we can measure the energy of a physical system by performing a measurement of the Hamiltonian operator, which we defined in the equation (3.1). The equation for the energies is

$$Hf(x) = Ef(x).$$

After substituting our Hamiltonian and $E = k^2$, we get

$$-\frac{d^2}{dx^2} f(x) = k^2 f(x).$$

To solve this homogeneous differential equation with constant coefficients, we will substitute $f(x) = e^{\lambda x}$, where λ is a complex constant. Plugging this into the previous equation, we get

$$(\lambda^2 + k^2)e^{\lambda x} = 0.$$

From this, we get $\lambda = \pm ik$. The general solution of the equation is

$$f(x) = ae^{ikx} + be^{-ikx}.$$

Using De Moivre's formula we get

$$f(x) = A\cos(kx) + B\sin(kx), \tag{3.2}$$

where A a B are new complex constants. The derivative is

$$f'(x) = -Ak\sin(kx) + Bk\cos(kx). \tag{3.3}$$

This is where the solution starts to become specific depending on the graph we are working with.

First, we need to parameterize the edges as in which vertex the edge starts ($x = 0$) and where the edge ends ($x = l_j$, where l_j is the length of edge e_j). Then, we evaluate the vertex conditions at each vertex, substituting the general solution, which will yield us one or more equations. After doing this for every vertex, we will end up with a system of equations, which can be written as

$$MA = 0, \tag{3.4}$$

where A is a column vector of the complex constants and M is the system of equations rewritten as a matrix. This system has a non-trivial solution only if the determinant of matrix M is zero. From this condition, we can find the values for k and then the energy levels from $E = k^2$.[19]

As a lot of these conditions do not have an analytical solution, a numerical one is required. The operation is simple – find roots of a real function. To check that our root-finding solution is precise enough and we are not loosing too much roots, we will use Weyl's law. The equation

$$N(a, b) = \frac{\mathrm{L}}{\pi}(b - a) + O(1) \tag{3.5}$$

gives us an approximate number of roots in a given interval. L is the sum of edge lengths and $O(1)$ is the error. The meaning behind $O(1)$ is that the error should be limited by constants, implying that it should not rise when increasing the size of the interval (a, b). It should be noted that this approximation works only for large intervals. [18]

Chapter 4

Numerical methods for finding roots

In this chapter, we will discuss the importance of numerical methods for finding roots. This will be very useful later on, because doing the NND for quantum graphs involves a lot of root-finding. The content of this chapter will be mainly devoted to two methods – the bisection method and Brent's method. The last section will also contain a simple comparison of their speed and reliability.

4.1 Introduction

The task of finding a root or roots of a function is very simple in principle, as it only involves solving an equation

$$f(x) = 0.$$

The equation we are solving for can be either one-dimensional or have N dimensions. Finding roots in N dimensions proves to be significantly more challenging than in one dimension. The main difference is that in in N dimensions, is it not possible to bracket (or trap) a root between two bracketing values, because in N dimensions, there is not a certainty that the root is there until you have found it. The functions we will be working with in the following chapter are all one-dimensional.

The general idea behind root-finding algorithms is iteration. The routine, with each iteration, applies the chosen algorithm, which refines an initial estimate of the root. The algorithm iteratively improves the solution, until a certain level of accuracy has been met. These algorithms exhibit differences in terms of their speed and certainty in reaching the solution. The rate of convergence refers to the speed or efficiency at which an algorithm approaches the true root of a function as the number of iterations increases. It provides a quantitative measure of how quickly the approximations generated by the algorithm converge to the actual root. Unfortunately, the ones that are sure to converge are also the ones making the slowest progress when finding a root, so there is always a trade-off between speed and certainty of finding a solution. [20] We will now examine some root-finding routines.

4.2 Bisection method

The bisection method is one of the methods that cannot fail; it will always find a root, if the root is there. A root is bracketed in the interval (a, b) if $f(a)f(b) < 0$. Then, as per the intermediate value theorem (IVT), if the function is continuous, one or more roots must lie in that interval. [21] It is important to note that this applies only to continuous functions. Let us take a function

$$f(x) = \frac{1}{x}$$

for example. If we would take an interval $(-1, 1)$, then $f(-1)f(1) > 0$, but the root is not there, only singularity.

The bisection method is known for its linear convergence rate. This means that with each iteration, the bisection method approximately halves the interval that contains the root. The error between the approximation and the true root reduces by a factor of approximately $1/2$ in each iteration. Mathematically, the convergence rate can be expressed as $O(1/2^k)$, where k is the number of iterations. The workings of this algorithm can be seen in this Python implementation:

```
def bisection(f, a, b):
    # Set the tolerance
    tolerance = 10 ** -10

    # Check if the interval is within the tolerance
    if abs(a - b) < tolerance:
        return (a + b) / 2

    # Calculate the midpoint of the interval
    midpoint = (b - a) / 2

    # Determine which half of the interval to continue with
    if f(a + midpoint) * f(b) < 0:
        return bisection(f, a + midpoint, b)
    else:
        return bisection(f, a, b - midpoint)
```

In principle, this implementation is very straightforward. It halves the interval by either adding to the lower bound or subtracting from the upper bound. It then recursively continues until the solution is very close to the actual root. The only precondition for finding a solution is that $a < b$ and (a, b) contains a root. Also, the root remains bracketed for the entire duration of the evaluation. The maximum number n of iterations needed to achieve a required tolerance ϵ is

$$n = \log_2 \frac{\epsilon_0}{\epsilon},$$

where ϵ_0 is the size of the initial bracketing interval. This is the main motivation for using the bisection method, as no other method can guarantee a better worst case scenario.[22] Its average performance under standard standard assumptions is, however, sub-optimal, resulting in other methods usually being a better choice. [23]

4.3 Secant method

Many root-finding techniques employ the concept of interpolation to approximate the root of a function. Interpolation involves utilizing the last computed approximate values of the root to construct a polynomial of low degree that matches the function's values at those approximations. The root of this polynomial is then determined and used as a new approximation of the root of the function, and the process continues iteratively. [20]

The secant method is based on interpolating the function using two values, resulting in a polynomial of degree one, which approximates the function's graph as a line. This is an example of linear interpolation. This involves defining an auxiliary function that operates on the most recently computed approximations of a root, producing a new approximation as the output. This method is very similar to Newton's method, which is also an iterative method, but to use Newton's method, one must know the derivative of the function. This is, in our case, a deal breaker, because the functions we will be working with tend to be very large and computing their derivative is practically impossible. The basic idea of the Secant method is using a succession of roots of secant lines to better approximate a root of a function f. The method is defined by the recurrence relation

$$x_n = x_{n-1} - f(x_n - 1)\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}.$$

This formula requires two initial values x_0 and x_1, ideally chosen as close to the root as possible. [20] This method can be implemented in Python as show in in the following code snippet.

```
tolerance = 1e-10

# The auxiliary function that defines Secant method
def find_next_x(f, x0, x1):
    return x1 - (f(x1) * (x1 - x0) / f(x1) - f(x0))

def secant(f, x0, x1, number_of_iterations):
    if abs(x0 - x1) < tolerance:
        return x0

    # The method stops after a certain number of iterations
    # when it did not converge
    if number_of_iterations > 100:
        print("Failed to converge")
        return None

    x2 = find_next_x(f, x0, x1)
    return secant(f, x1, x2, number_of_iterations + 1)
```

Note that this and also the previous implementations use recursion. Same result can be achieved with iteration by introducing a while loop. The choice is based only on personal preference and readability.

The secant method has a convergence rate that is typically faster than linear but slower than quadratic. It converges at a rate close to the golden ratio, approximately 1.618, which is slower than the quadratic convergence seen in methods like Newton's method. The error

between the approximation and the true root reduces by a factor close to the golden ratio in each iteration. Mathematically, the convergence rate can be expressed as $O(\frac{(1+\sqrt{5})}{2^k})$, where k is the number of iterations. The secant method also has the advantage of requiring only one function evaluation every iteration, which can make a huge difference when dealing with large functions. The disadvantage is that this method is not very robust. Unlike the bisection method, it does not require the root to remain bracket, indicating it may not always converge. [20]

4.4 Brent's method

Brent's method is a hybrid root-finding algorithm that combines the advantages of several techniques, including the robustness of the bisection method and the superlinear convergence of the secant method and inverse quadratic interpolation. The key idea behind Brent's method is to dynamically select the most appropriate root-finding algorithm at each iteration, based on the behavior of the function and the convergence progress. It does this by bracketing the root and using the inverse quadratic interpolation to estimate the next root. While linear interpolation uses two points to estimate the function, inverse quadratic interpolation uses three points to fit an inverse quadratic function (where x is a quadratic function of y). Then, its value at $y = 0$ is taken as the next estimate of the root x. [24]

The implementation can be seen in the source code for the library scipy.optimize. [25]. In short, it attempts the inverse quadratic interpolation if possible. If not, it will fallback to the Secant method and if that decision would also not lead to a better estimate, it uses the bisection method as its last resort. This guarantees a linear convergence rate in the worst case scenario, but superlinear convergence rate on average. Brent also claims that this method will always converge as long as the values of the function are computable within a given region containing a root. [24]

We will now compare the difference in speed of the bisection method and Brent's method, specifically their SciPy [25] implementations. To do this, we will find roots of three distinct functions for the interval $(0, 10^5)$ first using the bisection method and then Brent's method. Then, we will compare the time it took them to find all the roots in the given interval. The result can be seen in Table 4.1. It is clear that Brent's method is faster than bisection, although not by a large margin. The difference is more noticeable with more complex (as in the opposite of simple) functions, which is the kind of functions we will be dealing with. It should also be noted that the number of found roots for each function was the same for both methods. For $\sin(x)$, it was 31 831, which is also the actual number of roots for this function in this interval, as per an equation $N = \frac{10^5}{\pi}$. For the other two functions, it was 379 403 and 767 698 respectively.

Method	f_1	f_2	f_3
Bisection	80.6 s	599 s	1622 s
Brent's method	79.8 s	547 s	1349 s

Table 4.1: Results of a root-finding methods benchmark. The interval for each benchmark was $(0, 10^5)$. $f_1 : sin(x), f_2 : \det(M)\ of\ (5.4), f_3 : (5.2)$.

Chapter 5

Nearest-neighbor distribution for certain graphs

In this chapter, we will do the NND for the Gaussian Ensembles (GOE, GUE and GSE). After that, the NND will be done for three quantum graphs, comparing the results with that from RMT. Most of the work here will be done in Python, with some parts using Wolfram Mathematica. The code for this chapter can be found on my GitHub repository: `https://github.com/stepazel/QuantumChaos`.

5.1 Gaussian Ensembles

To do the NND for the Gaussian Ensembles, we first need to sample the corresponding matrices. For that, we used Python library scikit-rmt by Alejandro Santorum (available at `https://github.com/AlejandroSantorum/scikit-rmt`), which has the algorithms for sampling GOE, GUE and GSE already implemented.

We start with checking that the Wigner's semicircle law holds. As shown in the figures below, for $n = 2$, where n are the dimensions of the matrix, the eigenvalues distribution shape is far from looking like a semicircle. By increasing the size of the matrix, we see the shape begin to take form of a semicircle, which is in correspondence with the law, which holds for $n \to \infty$. The difference between the larger dimensions is not as significant as for the smaller dimensions.

Now, we will perform the NND for GOE, but first, we will show why the unfolding procedure is needed. The NND will be done for matrices of size 2 and 50. We will also plot function (2.2), which is the Wigner's surmise for GOE. To perform the NND, we sample the GOE matrix, compute its eigenvalues, sort them and then get the individual differences. We also need to normalize the spacings and we do that by multiplying every value by $\frac{1}{\bar{s}}$, where $\bar{s}$ is the arithmetic mean of the spacings. For the smaller matrix, we see a clear agreement between Wigner's surmise and the NND. But when we go ahead and try do the same for a larger matrix, we start seeing a slight difference as the spacings tend to be on the smaller side (a shift to the left). The difference gets bigger for larger dimensions. So, in order to get rid of this shift and start examining larger matrices, we need to perform the unfolding procedure. To do this, we simply map the function (2.5) onto every value. It should be also noted that after the unfolding, $\bar{s}$ is equal to one, so normalizing the spacings is not needed.

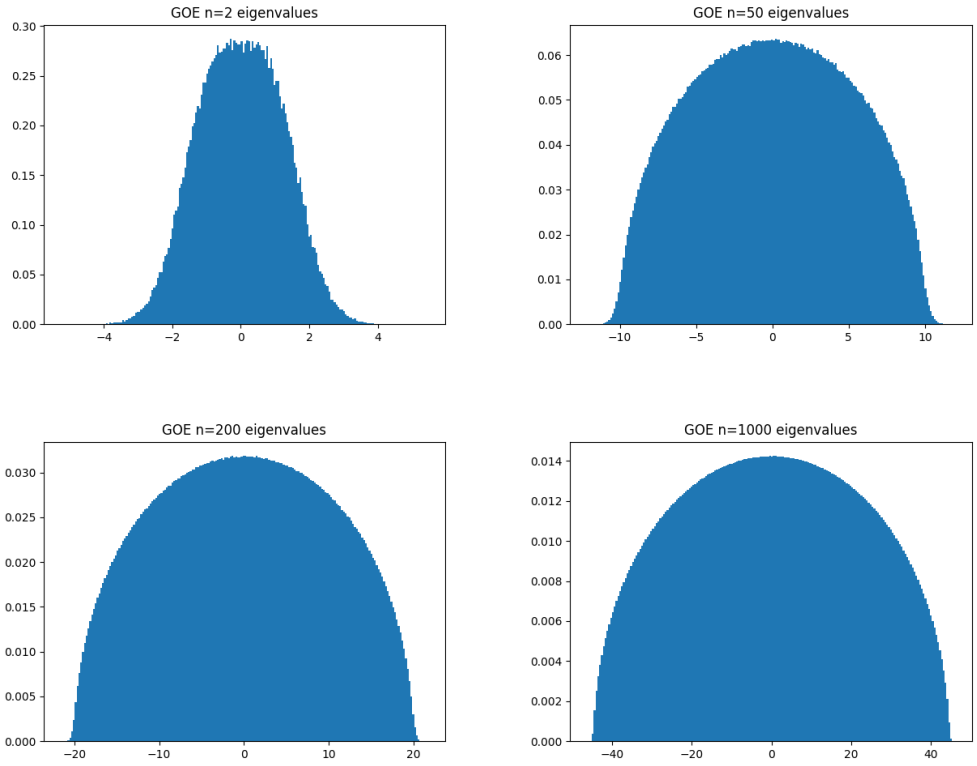

Figure 5.1: Semicircle law check. We clearly see that for larger dimensions, the distribution of eigenvalues takes on the semicircle shape.

Now that we know the unfolding is very much mandatory for larger dimensions, we can safely go ahead and do the NND for $n = 1000$. We will do this for all three matrices from the Gaussian Ensemble. The procedure stays the same, the only thing changing is the matrix sampled and the function plotted. For GOE, GUE and GSE we will plot (2.2), (2.3) and (2.4) respectively.

In Figure 5.4, we can see that for all of the matrices from the Gaussian Ensemble, even for larger dimensions, the Wigner's surmise holds.

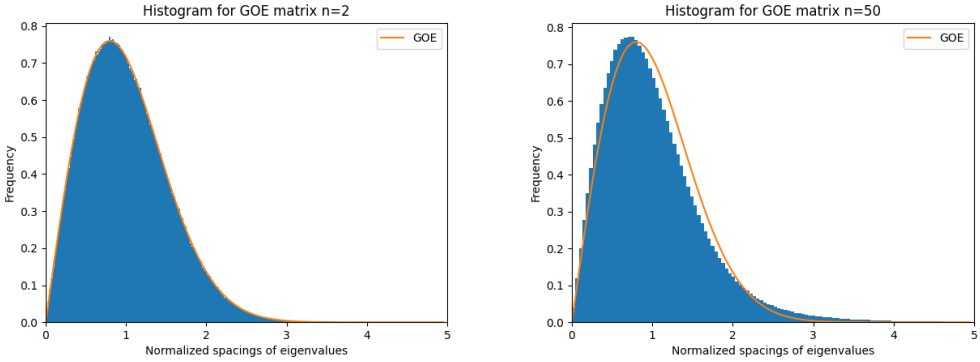

Figure 5.2: The spacings' size is shifted to the left for larger matrices.

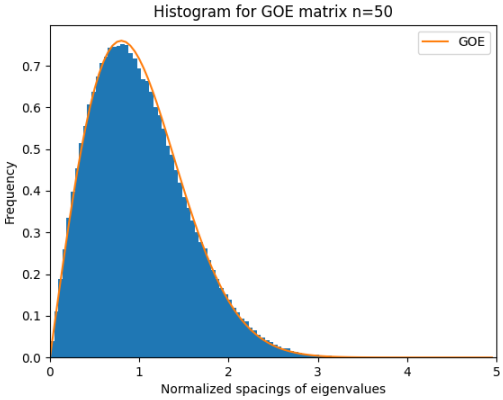

Figure 5.3: After the unfolding is done, an agreement with the Wigner's surmise is seen.

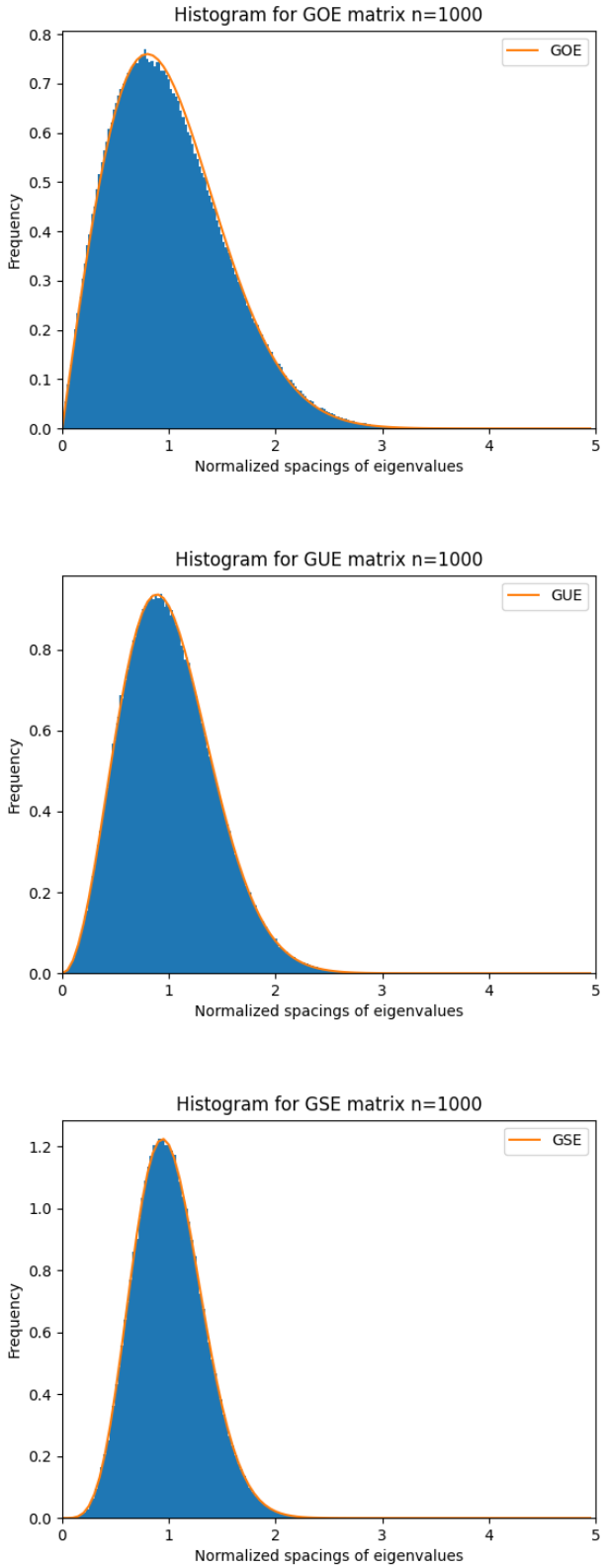

Figure 5.4: The NND for the Gaussian ensemble.

5.2 Star graph with Dirichlet conditions

Now, we will move on to some quantum graphs. We will start with a star graph that has Dirichlet condition imposed on every vertex. We parameterize the edges so that every edge starts on the outer vertex and they all end on the inner vertex. Because the Dirichlet conditions basically disconnect the edges, the graph should display a Poisson distribution similar to the distribution of random numbers, as seen in [18].

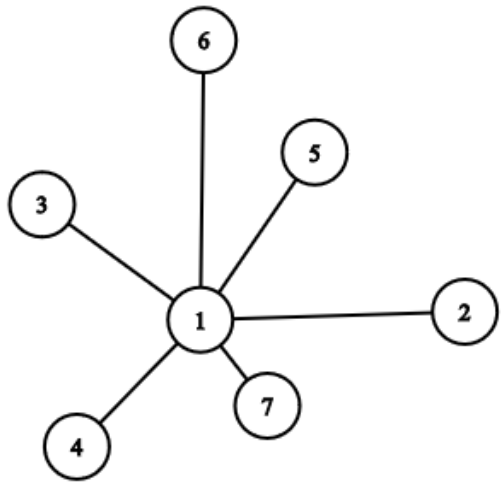

Figure 5.5: A star graph.

The Dirichlet condition gives

$$f_j(x) = 0$$

for each vertex. The equation (3.2) for the outer vertices is

$$f_j(0) = A_j,$$

which gives us $A_j = 0$. We then get

$$f_j(x) = B_j \sin(kx).$$

So, each edge gives us two equations; for the outer vertex it's $f_j(0) = 0$ and for the inner it's $f_j(x) = B_j \sin(kx)$. Rewriting this system of equations as a matrix gives us

$$\begin{pmatrix} \sin(k\nu_1) & 0 & \dots & 0 \\ 0 & \sin(k\nu_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sin(k\nu_j) \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_j \end{pmatrix} = 0, \tag{5.1}$$

where ν_j are edge lengths. The determinant of the first matrix is

$$\det M = \prod_j \sin(k\nu_j). \tag{5.2}$$

Now, before numerically finding the roots of this equation, we have to take into account two variables: the number of edges and their lengths. Neither of these can be chosen arbitrarily. For the edge lengths, we will impose two requirements:

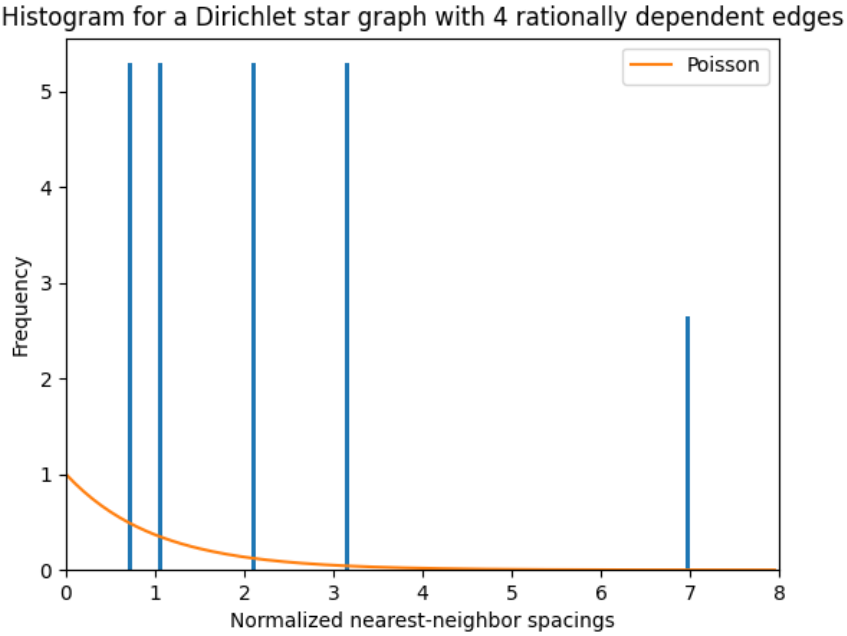

Figure 5.6: The NND for a star graph with six edges whose lengths are rationally dependent. The lengths are $(1, 2, 3, 4, 5, 6)$; a trivial example.

1. The edge lengths should be rationally independent. [18]
2. One edge length shouldn't be significantly greater than the others.

Our strategy to satisfy the first requirement will be choosing constants (π, e) and some numbers of our choice (e.g. 1.923) such that their ratios are far from rational numbers with small numerator and denominator. Lastly, we will apply functions like the square root or the natural logarithm. The demonstration of what happens with the result when we do not have a set of edge lengths that are rationally independent can be seen in Figure 5.6.

The second requirement is, at first glance, not very helpful, because the description significantly greater is very subjective. Is five significantly greater than one in the context of a quantum graph, or is it only greater? To put this requirement into context, let's plot the NND for a star graph with six edges, where one edge length is the cube of another. We can see the result in Figure 5.7. We can see a pattern where one spacing between the roots is repeating. This is greatly distorting the final result.

For this certain graph, a higher number of edges leads to better results, which is shown in Figure 5.8. The difference is less significant when we get to higher number of edges.

Now, will all these requirements in mind, we can go ahead and compose a suitable star graph. It will have 12 edges and edge lengths as follows:

$$(\pi, \mathrm{e}, \sqrt{2}, \ln 2, 1.98712, \sqrt{5}, \sqrt{\mathrm{e}}, \ln 5, \sqrt{\pi}, 0.97127, \sqrt{3}, \ln 3, \ln \pi, 1.52321, 0.86127).$$

We can now substitute these values into the equation (5.2) and find roots of this equation. But first, before plotting the results, we will consult Weyl's law to see how many roots are we loosing. To do this, we simply find roots for some intervals, which are getting bigger in size. The important thing is that the number of lost roots does not rise dramatically with larger

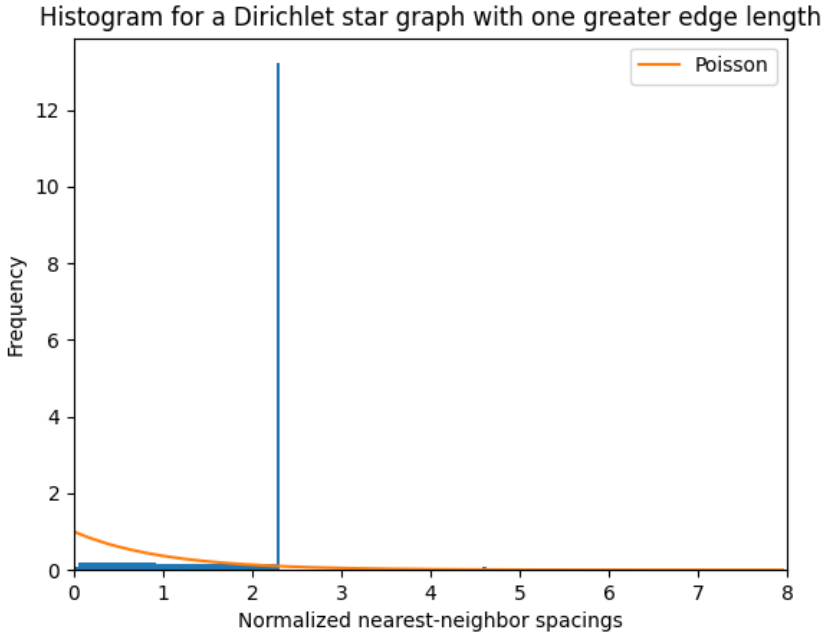

Figure 5.7: The edge lengths for this graph were $(\pi, \mathrm{e}, \sqrt{5}, \pi^3)$.

intervals. For this graph, we will find roots for intervals $[0, 2000], [0, 4000], [0, 6000], [0, 8000]$. Then, to find the number of lost roots, we will rearrange the equation (3.5) to give us the error as the difference between the approximate number of roots and the real number of roots like this

$$\frac{\mathrm{L}}{\pi}(b-a) - N(a,b) = O(1).$$

Interval size	2000	4000	6000	8000
Number of roots	15207	30562	45938	61268
Difference from Weyl's law	423	698	953	1253

Table 5.1: Results of Weyl's law for a star graph with Dirichlet conditions.

As we see in table Table 5.1, the results are not optimal. It seems like our root finding solution is missing a number of roots. A good thing is that this number does not rise drastically when increasing the size of the interval. Let us look at the histogram for the graph's NND.

The result in Figure 5.9 is what we expected; the NND has a distribution very similar to that of random numbers (i.e. Poisson distribution). This is also the result that Berkolaiko and Kuchment got in [18]. We can also see that our solution loses roots which are very close to another and the spacing between them approaches zero. Naturally, one would seek an improvement in increasing the resolution of the root-finding solution as in decreasing the size of the interval which a chosen root-finding algorithm takes as an input. But, after a certain threshold (i.e. a certain interval size), decreasing the interval size does not yield any better results. For this graph, a bisection method was used. Although there are other methods that converge faster, the bisection method was chosen because it's very robust and

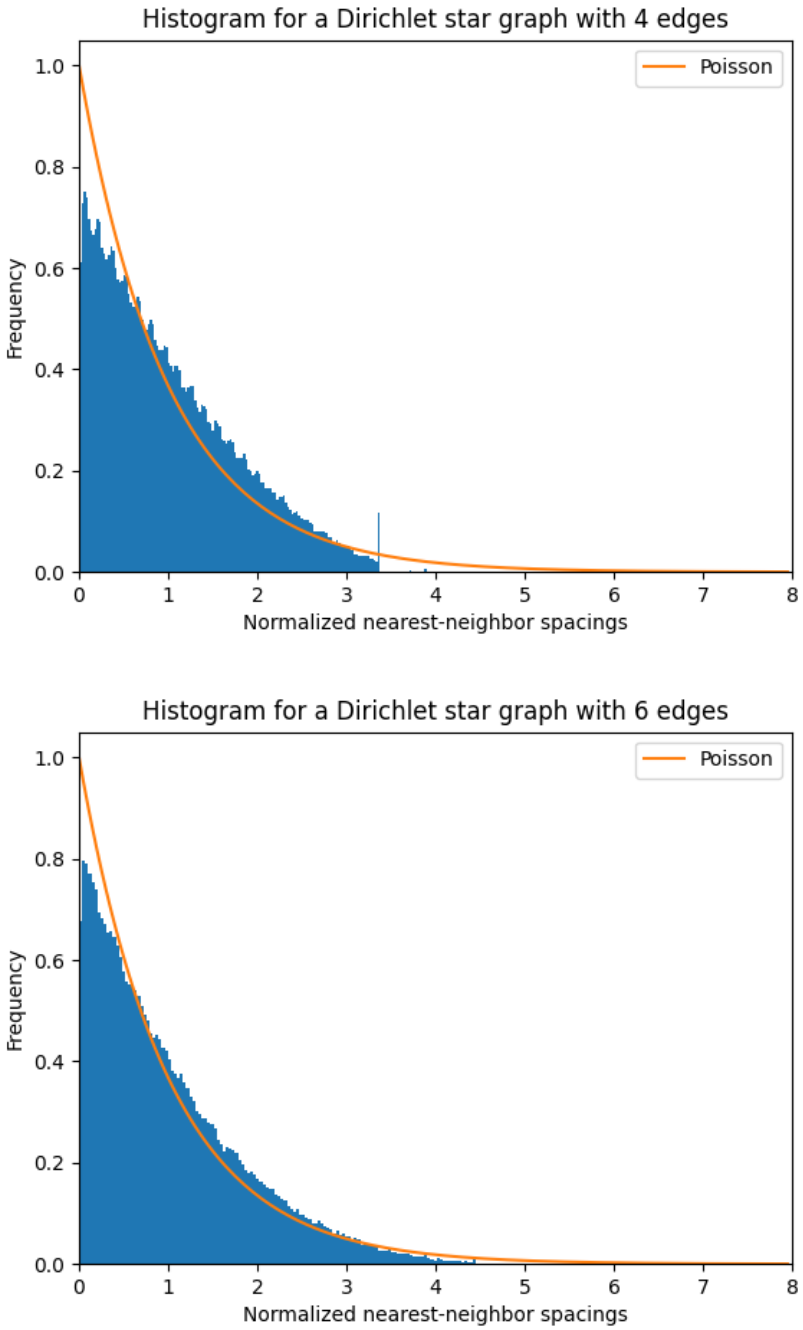

Figure 5.8: By increasing the number of edges from four to six, we achieve a better result. Edge lengths for the first graph are $(\pi, \mathrm{e}, \sqrt{5}, 2.5)$. For the second graph, we appended $(\ln 5, \sqrt{\pi})$.

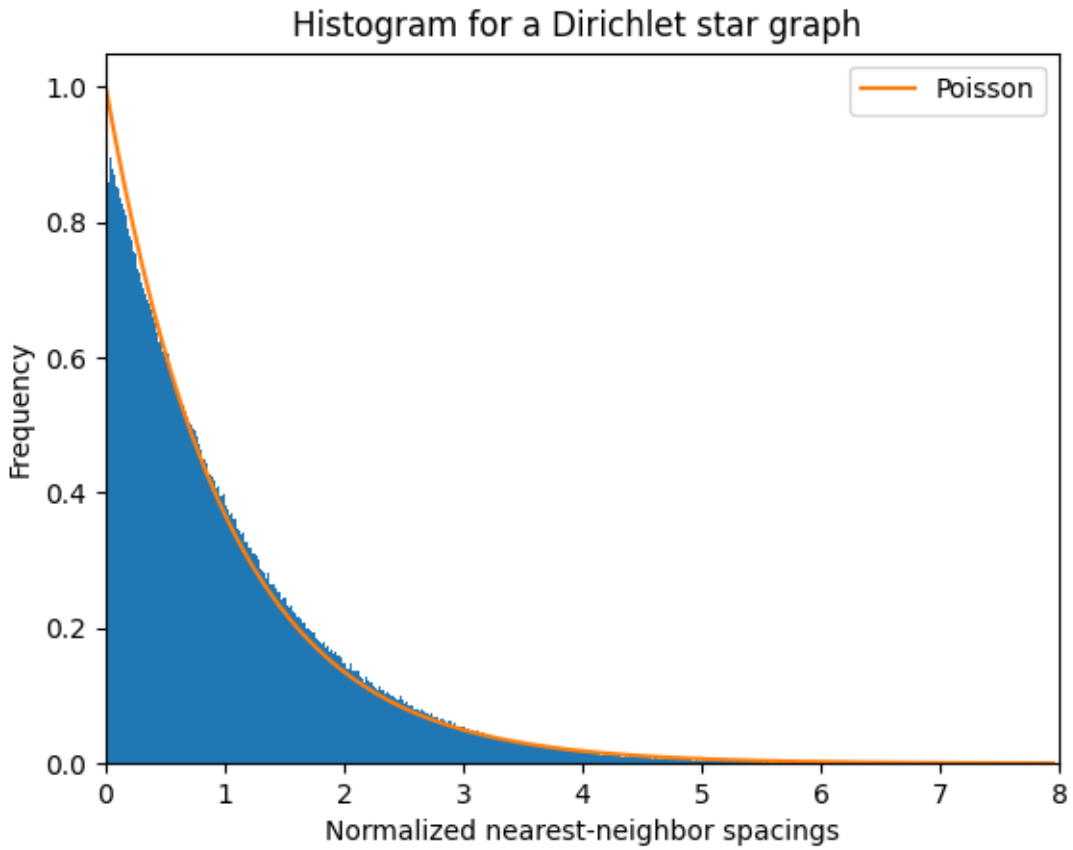

Figure 5.9: The NND histogram for star graph with Dirichlet conditions. The result is for 10^6 roots and 12 edges with lengths $(\pi, \mathrm{e}, \sqrt{2}, \ln 2, 1.98712, \sqrt{5}, \sqrt{\mathrm{e}}, \ln 5, \sqrt{\pi}, 0.97127, \sqrt{3}, \ln 3, \ln \pi, 1.52321, 0.86127)$.

in a given interval, it always finds a root (if the interval has one). Also, the equation we are solving (determinant of the (5.1) matrix) for is fairly simple, so speed is not an issue here. This is because the root-finding algorithm needs to evaluate the function a certain number of times to find the root and this evaluation takes significantly more time for more complex functions.

5.3 Star graph with Standard and Dirichlet conditions

We will continue with a star graph that has Dirichlet conditions imposed on its outer vertices and Standard condition on its center vertex. We parameterize the edges same as before, with $x = \nu_j$ in the center vertex. The Dirichlet conditions gives

$$f_j(x) = 0$$

for each outer vertex. The standard condition gives

$$\begin{aligned} f_1(\nu_1) = f_2(\nu_2) = \cdots = f_j(\nu_j), \\ -f_1'(\nu_1) - f_2'(\nu_2) - \ldots - f_j'(\nu_j) = 0. \end{aligned} \tag{5.3}$$

From the Dirichlet condition, we get

$$A_j \cos(0) + B_j \sin(0) = 0,$$

which gives us

$$f_j(x) = B_j \sin(kx),$$
$$f_j'(x) = kB_j \cos(kx).$$

Plugging in this result into (5.3), rewriting this system of equations as matrix and diving the last row by $-k$ gives us

$$\begin{pmatrix} \sin(k\nu_1) & -\sin(k\nu_2) & 0 & \dots & 0 \\ \sin(k\nu_1) & 0 & -\sin(k\nu_3) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sin(k\nu_1) & 0 & 0 & \dots & \sin(k\nu_j) \\ \cos(k\nu_1) & \cos(k\nu_2) & \dots & \cos(k\nu_{j-1}) & \cos(k\nu_j) \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_j \end{pmatrix} = 0. \tag{5.4}$$

The determinant of the first matrix in (5.4) is a sum of $\cos(k\nu_j)$ for each edge multiplied by sines of the remaining edge lengths. For a star graph with three edges, it would look like this:

$$\cos(k\nu_1)\sin(k\nu_2)\sin(k\nu_3) + \cos(k\nu_2)\sin(k\nu_3)\sin(k\nu_1) + \cos(k\nu_3)\sin(k\nu_1)\sin(k\nu_2).$$

Our graph will have six edges, or seven vertices respectively. The edge lengths are

$$(\pi, \mathrm{e}, \sqrt{2}, \sqrt{3}, \sqrt{\mathrm{e}}, 1.2754).$$

These numbers follow the same requirements as in the graph with only Dirichlet conditions, although the number of edges is not as important. Also, for this graph, we will use Brent's method. It is generally considered one of the best root-finding algorithms. For this function (and for many others), it is also faster then the previously used bisection method. We can

Interval size	4000	8000	12000	16000	20000
Number of roots	15176	30353	45529	60707	75885
Difference from Weyl's law	1.3	1.7	3	2.4	1.7

Table 5.2: Results of Weyl's law for a star graph with Dirichlet conditions on outer vertices and a Standard condition on the center vertex.

now use Weyl's law to find out how many roots are we losing. We will find roots in intervals [0, 4000], [0, 8000], [0, 12000], [0, 16000] and [0, 20000]. As we can see in Table 5.2, the result is very satisfying. We are loosing a very small number in roots, meaning we have got a very high-quality result.

In Figure 5.10, the NND almost, but not quite resembles GOE's distribution. The distribution is skewed to the right, indicating spacings of slightly larger size are more prevalent than needed to fit in the GOE's distribution. But overall, this results shows a decent correlation with the characteristic of a random matrix, implying this specific star graph demonstrates chaotic behavior.

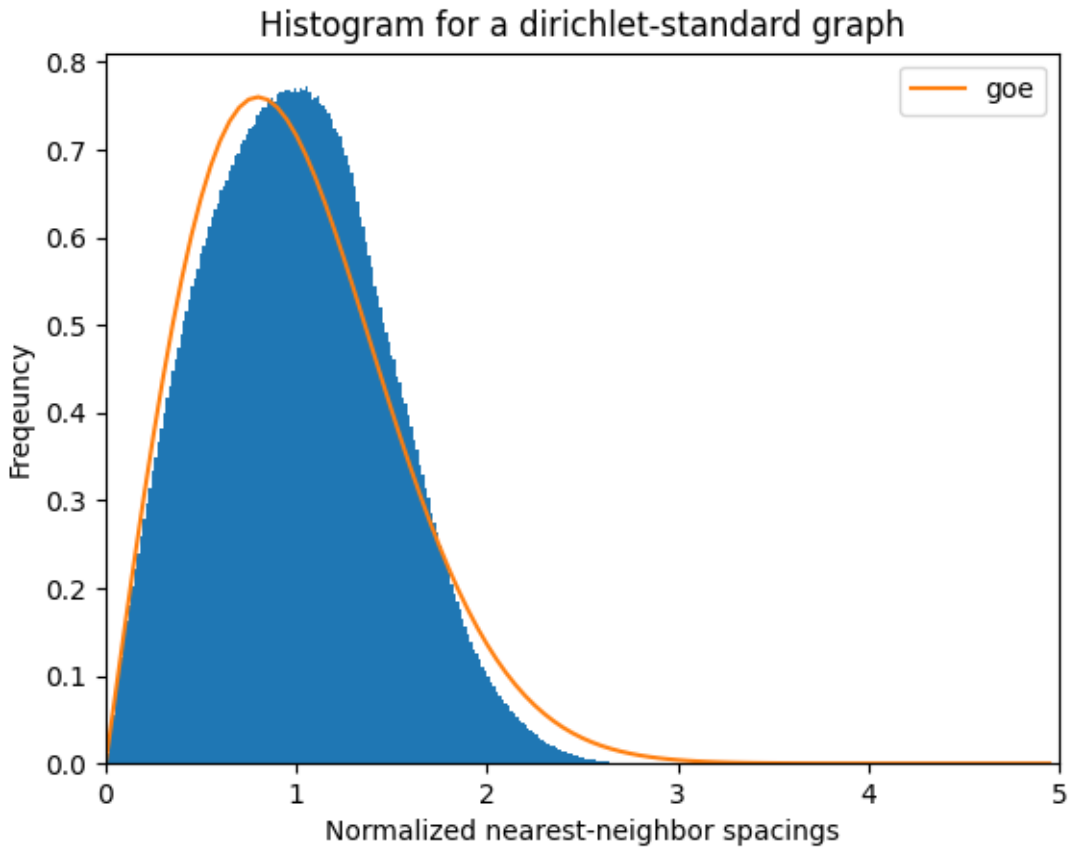

Figure 5.10: The NND for a star graph with 6 edges, Dirichlet conditions on the outer vertices and Standard condition on the center vertex. The result is for 10^7 roots.

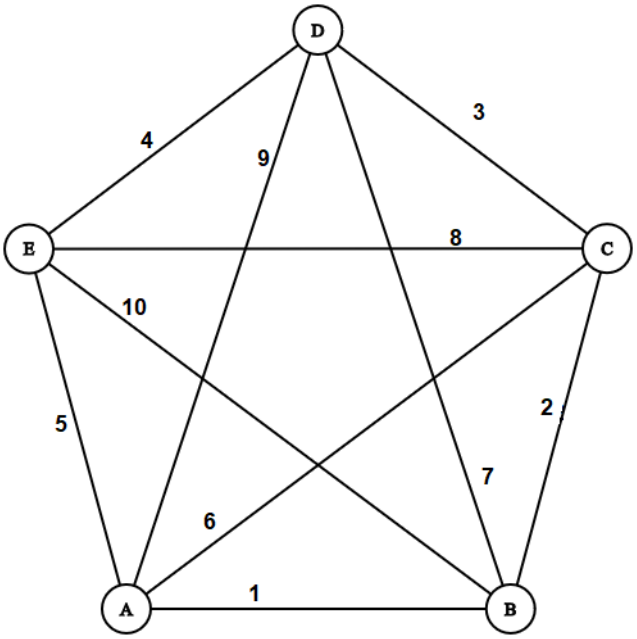

Figure 5.11: Complete graph with 5 vertices.

5.4 Complete graph with Standard conditions

In graph theory, complete graph is a graph in which all the vertices are connected with each other. A complete graph with n vertices is denoted by K_n and has $\frac{n(n-1)}{2}$ edges. We will do the NND for K_5 with Standard condition on every vertex.

The edges are all labeled by starting in a vertex A, labeling the outer edges counterclockwise and continuing with the inner edges. The result can be seen in Figure 5.11. We will then parameterize the edges. The procedure has a similar counterclockwise motion. The result is in Table 5.3.

	1	2	3	4	5	6	7	8	9	10
A	0				L_5	0			L_9	
B	L_1	0					0			L_{10}
C		L_2	0			L_6		0		
D			L_3	0			L_7		0	
E				L_4	0			L_8		0

Table 5.3: The parameterization of a K5 graph. L_j is the edge's length.

We will be finding the secular determinant of a 20×20 matrix. This is because the Standard condition gives four equations for each vertex. For vertex A, with our parameterization in place, the Standard condition gives

$$\begin{aligned} &f_1(0) = f_5(L_5) = f_6(0) = f_9(L_9), \\ &f_1'(0) - f_5'(L_5) + f_6'(0) - f_9'(L_9) = 0. \end{aligned} \tag{5.5}$$

After substituting (3.2) and (3.3) into (5.5), we get

$$\begin{aligned}
A_1\cos(0) + B_1\sin(0) - A_5\cos(kL_5) - B_5\sin(kL_5) &= 0,\\
A_1\cos(0) + B_1\sin(0) - A_6\cos(kL_6) - B_6\sin(kL_6) &= 0,\\
A_1\cos(0) + B_1\sin(0) - A_9\cos(kL_9) - B_9\sin(kL_9) &= 0,\\
-kA_1\sin(0) + kB_1\cos(0) + kA_5\sin(kL_5) - kB_5\cos(kL_5) -&\\
-kA_6\sin(0) + kB_6\cos(0) + kB_9\sin(kL_9) - kB_9\cos(kL_9) &= 0,
\end{aligned}$$

which make up first four rows of our desired matrix. Similar procedure can be employed with the remaining vertices. We will then rewrite the system of equations as (3.4) and find the determinant of M. Because of the dimensions of this matrix, analytical solution is not very suitable. We will use Wolfram Mathematica instead of Python for this task, because Mathematica's solution proved to be much faster. The edge lengths for this graph are

$$(\pi, \mathrm{e}, \sqrt{2}, \sqrt(3), \sqrt{e}, 1.2654, 0.5, \sqrt[4]{2}, \pi^2, \mathrm{e}^3).$$

Because the determinant of this matrix is a very long function, finding its roots is a lot slower, because the evaluation of this function is much more complex and time-consuming. Because of this, our histogram will not be as smooth as the previous ones.

There differences from Weyl's law in Table 5.4 are negative, as opposed to the previous results. This implies that we are actually finding roots that are not there. This may be a result of the function values being very small (e.g. 10^{-200}) and the root-finding algorithm interprets this as a root. Also, when analyzing the found roots, it was not uncommon to see two or more roots that are exactly one iteration step away, for example roots $1.005, 1.01, 1.015$, which also indicates a fault in the root-finding solution. However, even after decreasing the tolerance to 10^{-250}, the result is still the same.

In Figure 5.12, we can see a similarity with GOE's distribution, but it is still not quite right. Spacings in size between 0 and 1/2 are a little more prevalent then the ones with size in between 1/2 and 3/2. There is also a disagreement with Weyl's law, as seen in Table 5.4.

Interval size	25	50	75	100	125
Number of roots	183	371	559	750	939
Difference from Weyl's law	-25	-57	-88	-121	-153

Table 5.4: Results of Weyl's law for a K5 graph with Standard conditions.

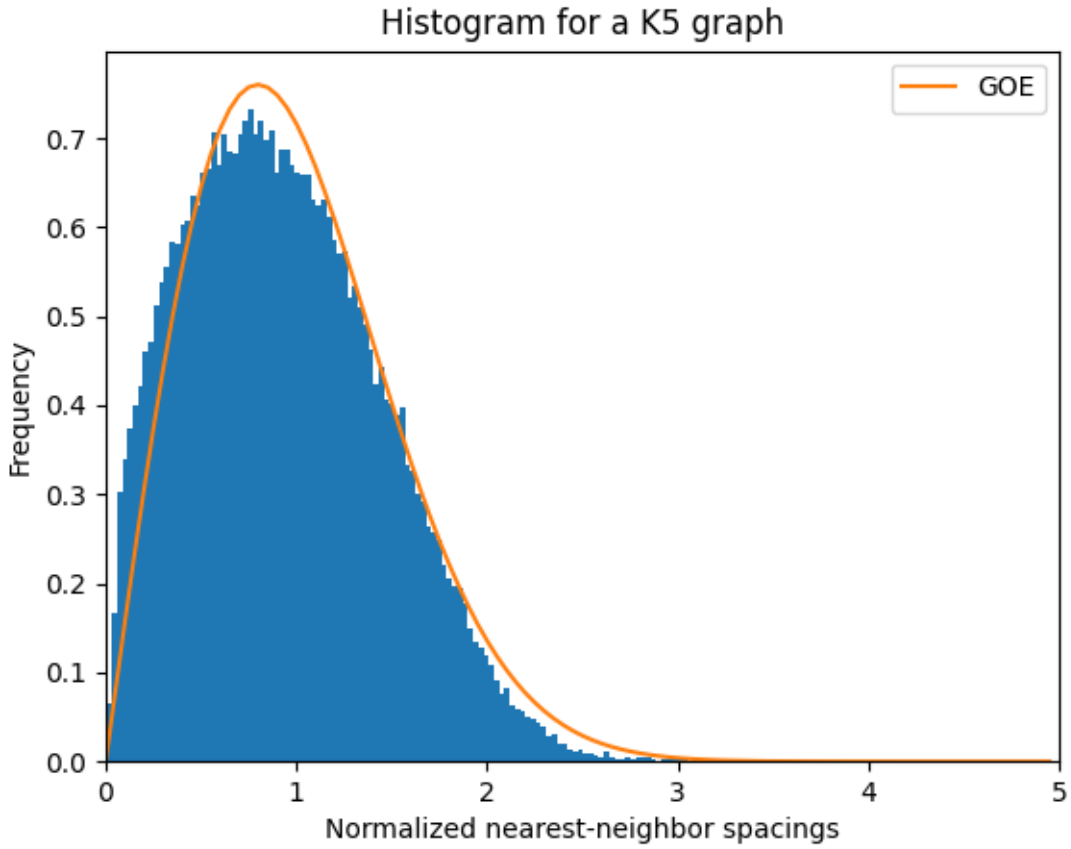

Figure 5.12: The NND for a K5 graph with edge lengths $(\pi, \mathrm{e}, \sqrt{2}, \sqrt{3}, \sqrt{e}, 1.2654, \sqrt[4]{2}, \pi^2, \mathrm{e}^3)$. The result is for 10_5 roots.

Conclusion

In the first chapter, we introduced and defined the basics of the issue of Quantum chaos and the methods that are associated with its study. In the next two chapters, we defined the concepts and models used in finding the NND of random matrices and quantum graphs. In the fourth chapter, we dealt with the issue of finding roots to justify the choice of a suitable root-finding algorithm for our needs.

In the last chapter, we have shown that Wigner's semicircle law (2.1) holds. We then computed the NND of the unfolded spectra of GOE, GUE and GSE. This result strongly aligns with the theoretical predictions given by Wigner's surmise in (2.2), (2.3) and (2.4).

We then moved on to computing the NND for certain quantum graphs. The result for the star graph with Dirichlet conditions imposed on its every vertex is in a fairly decent agreement with the prediction, although a considerable amount of roots was lost. Also, in this section, the motivation for choosing the optimal set of edge lengths for a quantum graph was demonstrated.

The star graph, characterized by Dirichlet conditions on its outer vertices and a Standard condition on its central vertex, exhibits a NND that closely resembles the distribution of the GOE, but with a noticeable shift to the right. Despite this deviation, the analysis remains credible since virtually no roots are lost in the process. This finding suggests that the combination of Dirichlet and Standard conditions induces interesting spectral behavior, where the Standard condition in the center vertex virtually connected the otherwise disconnected vertices, which leads to a more chaotic characteristics of its spectra.

The investigation of the NND for a K5 graph with Standard conditions on every vertex has yielded intriguing findings, although the result falls short of being deemed optimal. While the NND displays some similarities to the distribution observed in the GOE, it does not perfectly match GOE's characteristic shape. Specifically, the peak of the distribution does not attain its expected height, indicating a certain degree of deviation from GOE-like behavior. Moreover, the application of our root-finding solution introduced some distortions in the analysis. The solution identified more roots than anticipated (as seen in Table 5.4, leading to an altered NND. Further investigations and refinements of the root-finding method are necessary to obtain more accurate and robust results.

Bibliography

[1] Kottos, T.; Smilansky, U. Quantum Chaos on Graphs. *Physical Review Letters*, volume 79, no. 24, 1997: pp. 4794–4797, ISSN 0031-9007, doi:10.1103/PhysRevLett.79.4794.

[2] Bohigas, O.; Giannoni, M. J.; Schmit, C. Characterization of chaotic quantum spectra and universality of level fluctuation laws. *Physical Review Letters*, volume 52, 1984: pp. 1–4, doi:10.1103/PhysRevLett.52.1.

[3] Strogatz, S. H. *Nonlinear Dynamics and Chaos*. Boca Raton, Florida: Taylor & Francis, second edition, 2015, ISBN 9780429492563.

[4] Griffiths, D. J. *Introduction to Quantum Mechanics*. Shaftesbury Road, Cambridge: Cambridge University Press, fourth edition, 2018, ISBN 9781316995433.

[5] Miller, D. A. B. *Quantum Mechanics for Scientists and Engineers*. Shaftesbury Road, Cambridge: Cambridge University Press, 2008, ISBN 9780521897839, 161-199 pp.

[6] Al-Khalili, J. *Quantum: a guide for the perplexed*. Weidenfeld and Nicolson, 2003, ISBN 9781841882383, 104-5 pp.

[7] Berry, M. V. Semiclassical Mechanics of regular and irregular motion. In *Les Houches Lecture Series Session XXXVI*, edited by G. Iooss; R. H. G. Helleman; R. Stora, Amsterdam: North Holland, 1983, pp. 171–271.

[8] Berry, M. V. Quantum scars of classical closed orbits in phase space. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, volume 423, no. 1864, 1989: pp. 219–231.

[9] Heller, E. J. Bound-State Eigenfunctions of Classically Chaotic Hamiltonian Systems: Scars of Periodic Orbits. *Physical Review Letters*, volume 53, no. 16, Oct 1984: pp. 1515–1518.

[10] Luitz, D. J.; Laflorencie, N.; Alet, F. Extended slow dynamical regimes close to the many-body localization transition. *Physical Review B*, volume 93, no. 6, 2016: p. 060201.

[11] Haake, F. *Quantum Signatures of Chaos*. New York, NY: Springer Science & Business Media, 2010, ISBN 9783642054280, 45-59 pp.

[12] Wigner, E. P. Characteristic Vectors of Bordered Matrices with Infinite Dimensions. *Annals of Mathematics*, volume 62, no. 3, 1955: pp. 548–564.

[13] Mehta, M. L. *Random Matrices*. San Diego, USA: Elsevier, 2004, ISBN 9780080474113.

[14] Wigner, E. On the Distribution of the Roots of Certain Symmetric Matrices. *Annals of Mathematics*, volume 67, no. 2, 1958: pp. 325–328.

[15] Dyson, F. J. Statistical theory of the energy levels of complex systems. I. *Journal of Mathematical Physics*, volume 3, no. 1, 1962: pp. 140–156.

[16] Guhr, T.; Müller-Groeling, A.; Weidenmüller, H. A. Random matrix theories in quantum physics: common concepts. *Physics Reports*, volume 299, no. 4, 1998: pp. 189–425.

[17] Sweeney-Blanco, R. Computational Random Matrix Theory. `https://robertsweeneyblanco.github.io/Computational_Random_Matrix_Theory/Eigenvalues/Wigner_Surmise.html#Unfolding-of-Spectra`, n.d., accessed April 21, 2023.

[18] Berkolaiko, G.; Kuchment, P. *Introduction to Quantum Graphs*. Providence, Rhoda Island: American Mathematical Society, 2013, ISBN 978-0-8218-9211-4.

[19] Berkolaiko, G. An elementary introduction to quantum graphs, 2016, preprint. Available from: `https://arxiv.org/abs/1603.07356`

[20] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; et al. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, third edition, 2007, ISBN 978-0-521-88068-8.

[21] Cates, D. M. *Cauchy's Calcul Infinitésimal*. Berlin: Springer, 2019, ISBN 978-3-030-11035-2, 249 pp.

[22] Sikorski, K. Bisection is optimal. *Numerical Mathematics*, volume 40, 1982: pp. 111–117, doi:10.1007/BF01459080. Available from: `https://doi.org/10.1007/BF01459080`

[23] Graf, S.; Novak, E.; Papageorgiou, A. Bisection is not optimal on the average. *Numerische Mathematik*, volume 55, no. 4, 07 1989: pp. 481–491, ISSN 0945-3245, doi:10.1007/BF01396051.

[24] Brent, R. P. *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall, 1973, ISBN 0-13-022335-2.

[25] SciPy Developers. SciPy/Scipy.optimize: Optimization and Root Finding. GitHub repository, 2023. Available from: `https://github.com/scipy/scipy/tree/main/scipy/optimize`

List of Abbreviations

BGS Bohigas-Giannoni-Schmit

GOE Gaussian Orthogonal Ensemble

GSE Gaussian Symplectic Ensemble

GUE Gaussian Unitary Ensemble

NND nearest neighbor distribution

PDF probability density function

RMT Random Matrix Theory

List of Figures

List of Tables

Appendices

Appendix A

Code Listings

```
from numpy import pi, e


def goe(x):
    return (pi / 2) * x * (e ** ((-pi / 4) * x ** 2))


def gue(x):
    return (32 / pi ** 2) * x ** 2 * (e ** ((-4 / pi) * x ** 2))


def gse(x):
    return ((2 ** 18) / (3 ** 6 * pi ** 3)) * x ** 4 * (
               e ** ((-64 / (9 * pi)) * x ** 2))


def poisson(x):
    return e ** (-x)


def normalize_root(root: float, edge_lengths_sum: float) -> float:
    return (root * edge_lengths_sum) / pi
```

Listing A.1: Python file containing the functions for GOE, GUE, GSE and Poisson distributions. Also, a function to normalize the roots of a quantum graph is included.

```
import numpy as np
import matplotlib.pyplot as plt
from skrmt.ensemble import GaussianEnsemble

labels = {
    1: "GOE",
    2: "GUE",
    4: "GSE",
}

beta = 1
n = 1000
reps = 100_000

eigenvalues = []
for r in range(reps):
    matrix = GaussianEnsemble(beta=beta, n=n,
                              use_tridiagonal=True).matrix
    sorted_eigen_values = np.sort(np.linalg.eigvalsh(matrix))
    eigenvalues.extend(sorted_eigen_values)
    if r % 10_000 == 0:
        print(r)

plt.title(f'{labels[beta]} n={n} eigenvalues')
plt.hist(eigenvalues, bins=200, density=True)
plt.savefig(f'{labels[beta]} n={n} eigenvalues')
plt.show()
```

Listing A.2: Python script to compute the eigenvalues of a matrix in the Gaussian ensemble and display them as a histogram.

```
from typing import Callable
from scipy.optimize import root_scalar
import time

tol = 1e-12


def find_roots(func: Callable, number_of_roots: int, precision=400,
               method: str = 'brentq') -> list:
    roots = []
    i = 0
    tic = time.perf_counter()
    percentage = 0

    while len(roots) < number_of_roots:
        a = i / precision
        b = a + 1 / precision
        interval = [a, b]

        if func(a) * func(b) > 0:
            i += 1
            continue

        result = root_scalar(func, bracket=interval, xtol=tol,
                             method=method)
        if result.converged and result.root not in roots:
            roots.append(result.root)
        i += 1
        if len(roots) % (number_of_roots / 100) == 0 and len(
                roots) != 0:
            print(f"{percentage}% completed")
            percentage += 1
    toc = time.perf_counter()
    print(f"Elapsed time {toc - tic:0.4f} seconds")
    print(f"Ending was at {i / precision}")
    return roots
```

Listing A.3: Python function to find a given number of roots of an arbitrary function.

```
from sympy import Matrix
from sympy.parsing.mathematica import parse_mathematica
from sympy import lambdify
from sympy import sympify
from sympy import Symbol
from numpy import pi, e


def convert_to_mathematica_matrix(matrix: Matrix) -> str:
    """ Converts sympy matrix into a string that can be pasted into Mathematica input.
    matrix_str = str(matrix.tolist())
    return matrix_str \
        .replace('[', '{') \
        .replace(']', '}') \
        .replace('cos', 'Cos') \
        .replace('sin', 'Sin') \
        .replace('(', '[') \
        .replace(')', ']')


def get_func_from_mathematica(file_name: str):
    with open(file_name, 'r') as file:
        content = file.read()

    expr_str = parse_mathematica(content)

    # Convert the string to a sympy expression
    expr = sympify(expr_str)

    # Define the variables
    k = Symbol('k')
    # Use lambdify to create a function that can be evaluated numerically
    subbed_expr = expr.subs({'pi': pi, 'e': e})
    func = lambdify(k, subbed_expr, 'numpy')
    return func
```

Listing A.4: Python file with a function that converts a Python matrix to a Mathematica matrix, which can then be pasted into Mathematica program. The second function does the opposite.

```
import time
import numpy as np
from skrmt.ensemble import GaussianEnsemble
import matplotlib.pyplot as plt
from distributions import goe, gue, gse
from unfolding import F

functions = {
    1: ("GOE", goe),
    2: ("GUE", gue),
    4: ("GSE", gse),
}

beta = 1
n = 1000
reps = 1000
spacings = []

tic = time.perf_counter()
for r in range(reps):
    matrix = GaussianEnsemble(beta=beta, n=n,
                              use_tridiagonal=True).matrix
    sorted_eigen_values = np.sort(np.linalg.eigvalsh(matrix))
    unfolded_values = [F(eigenvalue, n, beta) for eigenvalue in
                       sorted_eigen_values]
    spacings.extend(np.diff(unfolded_values))

mean = np.mean(spacings)
normalized_spacings = list(map(lambda s: s / mean, spacings))
print(mean)
toc = time.perf_counter()
print(f'Elapsed time: {toc - tic:0.4f} seconds')

np.save(f"{functions[beta][0]}{n}x{n} unfolded and normalized",
        normalized_spacings)

plt.hist(spacings, bins=100, density=True)
x = np.arange(0, 5, 0.05)
plt.plot(x, functions[beta][1](x), label=functions[beta][0])
plt.legend()
plt.xlabel("Normalized spacings of eigenvalues")
plt.ylabel("Frequency")
plt.xlim(xmin=0, xmax=5)
plt.title(f'Histogram for {functions[beta][0]} matrix n={n}')
plt.savefig(functions[beta][0])
plt.show()
```

Listing A.5: Python file that computes the NND for a matrix in the Guassian ensemble.

```
from sympy import symbols, integrate, sqrt, pi
import numpy as np

t = symbols("t")
N, Îš = symbols("N Îš", positive=True)
ÏČ = 2 / (pi * (2 * Îš * N)) * sqrt((2 * Îš * N) - t ** 2)
ÎŻ_N_Îš = integrate(N * ÏČ, t)

Îż = symbols("Îż")
F_Îż = ÎŻ_N_Îš.subs(t, Îż) - ÎŻ_N_Îš.subs(t, -sqrt(2 * Îš * N))


def F(eig, size, beta):
    if eig <= -np.sqrt(2 * beta * size):
        return 0
    if eig >= np.sqrt(2 * beta * size):
        return size
    else:
        return float(F_Îż.subs(Îš, beta).subs(N, size).subs(Îż, eig).evalf())
```

Listing A.6: Python file with an unfolding function F. Taken from [17].